CSc 466/566

Computer Security

18 : Network Security

Version: 2014/11/04 16:56:51

Department of Computer Science
University of Arizona

collberg@gmail.com

Christian Collberg

Outline

Tunneling

- TCP packets are sent in cleartext.
- In a tunneling protocol, packets are automatically encrypted.
- How to evade school firewall, secure traffic tunneling & more:
  http://www.wonderhowto.com/how-to-evade-school-firewall-secure-traffic-tunneling-more-285775
- Quick and Dirty SSH Tunneling: http://www.youtube.com/watch?v=EkPzt4_S3cc

SSH Protocol

1. Client connects to Server over TCP.

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
    1. Password:

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
    1. Password:
        - The Client sends the Server their password.

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
    1. Password:
        - The Client sends the Server their password.
    2. Public-key:

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
    1. Password:
        - The Client sends the Server their password.
    2. Public-key:
        1. The Client sends the Server their public key P_{client}.

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
   1. Password:
      - The Client sends the Server their password.
   2. Public-key:
      1. The Client sends the Server their public key P_{client}.
      2. The Server checks if P_{client} is in its database.

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
   1. Password:
      - The Client sends the Server their password.
   2. Public-key:
      1. The Client sends the Server their public key P_{client}.
      2. The Server checks if P_{client} is in its database.
      3. The Server generates a challenge challenge.

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
   1. Password:
      - The Client sends the Server their password.
   2. Public-key:
      1. The Client sends the Server their public key P_{client}.
      2. The Server checks if P_{client} is in its database.
      3. The Server generates a challenge challenge.
      4. The Server sends $E_{P_{\text{client}}}(\text{challenge})$ to the Client.

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
   1. Password:
      - The Client sends the Server their password.
   2. Public-key:
      1. The Client sends the Server their public key P_{client}.
      2. The Server checks if P_{client} is in its database.
      3. The Server generates a challenge challenge.
      4. The Server sends $E_{P_{\text{client}}}(\text{challenge})$ to the Client.
      5. The Client decrypts: $D_{S_{\text{client}}}(E_{P_{\text{client}}}(\text{challenge}))$

SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
   1. Password:
      - The Client sends the Server their password.
   2. Public-key:
      1. The Client sends the Server their public key P_{client}.
      2. The Server checks if P_{client} is in its database.
      3. The Server generates a challenge challenge.
      4. The Server sends $E_{P_{\text{client}}}(\text{challenge})$ to the Client.
      5. The Client decrypts: $D_{S_{\text{client}}}(E_{P_{\text{client}}}(\text{challenge}))$
      6. The Client sends challenge to the Server.

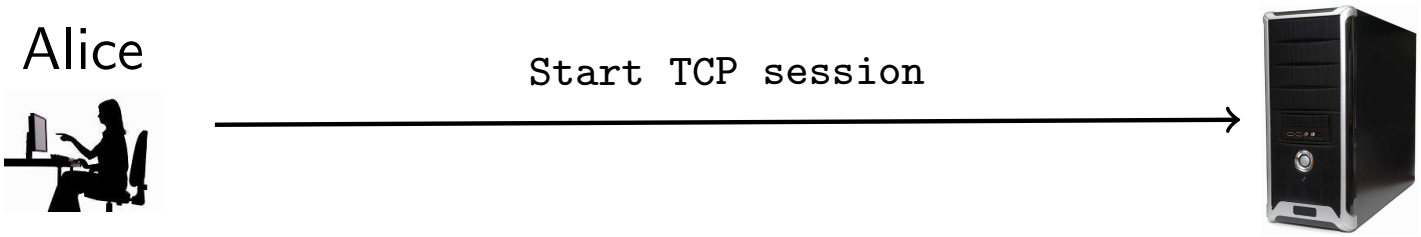
SSH Protocol

1. Client connects to Server over TCP.
2. Client and Server exchange available cryptographic primitives.
3. Client and Server decide on a shared session key.
4. The Server sends the Client a list of acceptable forms of authentication:
   1. Password:
      - The Client sends the Server their password.
   2. Public-key:
      1. The Client sends the Server their public key P_{client}.
      2. The Server checks if P_{client} is in its database.
      3. The Server generates a challenge challenge.
      4. The Server sends $E_{P_{\text{client}}}(\text{challenge})$ to the Client.
      5. The Client decrypts: $D_{S_{\text{client}}}(E_{P_{\text{client}}}(\text{challenge}))$
      6. The Client sends challenge to the Server.
5. The Server gives the Client a command prompt.

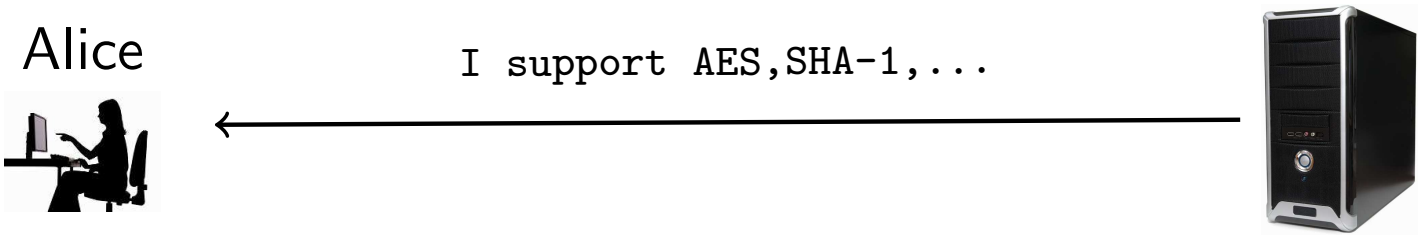
SSH Protocol

Alice

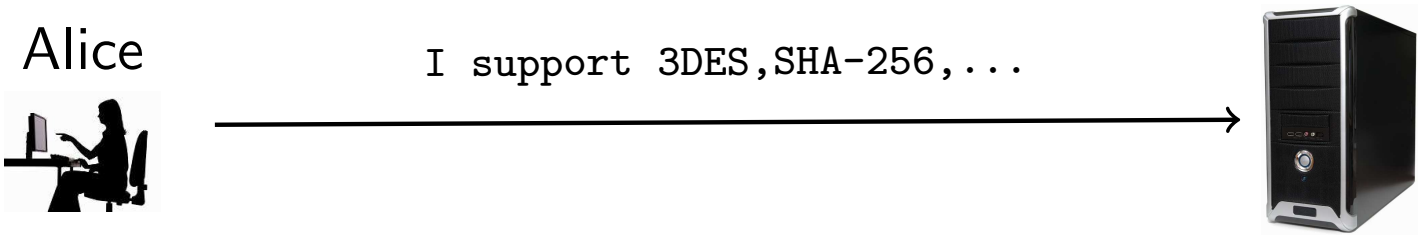
SSH Protocol



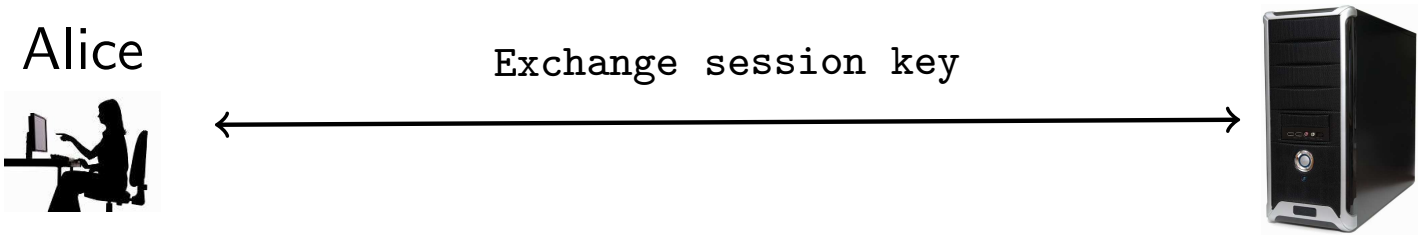
SSH Protocol



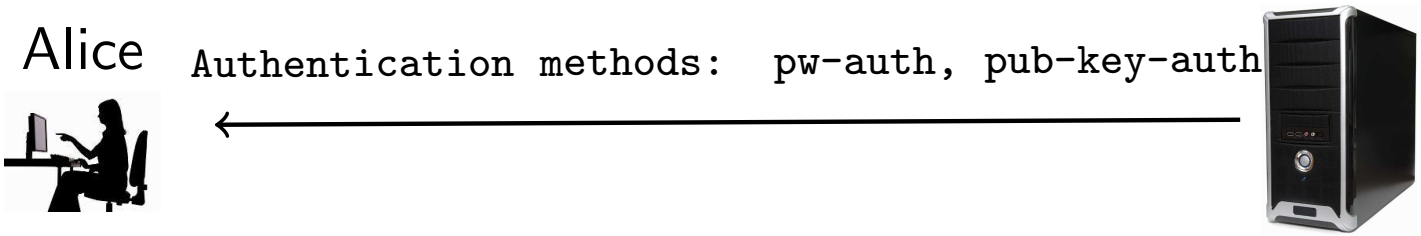
SSH Protocol



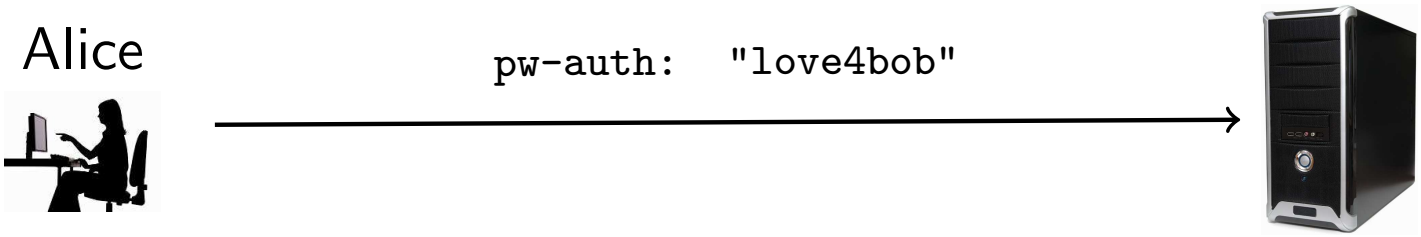
SSH Protocol



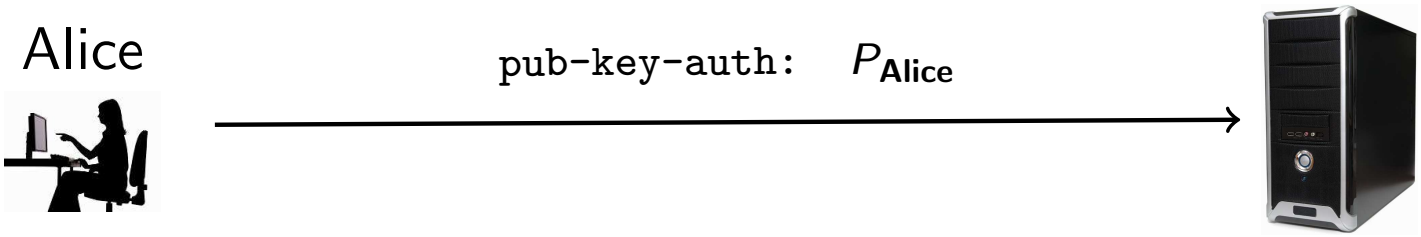
SSH Protocol



SSH Protocol



SSH Protocol



SSH Protocol

Alice

$P_{\mathbf{Alice}}$

user	$P_{\mathbf{user}}$
Alice	$P_{\mathbf{Alice}}$
Bob	$P_{\mathbf{Bob}}$

SSH Protocol

Alice

OK

user	$P_{\textbf{user}}$
Alice	$P_{\textbf{Alice}}$
Bob	$P_{\textbf{Bob}}$

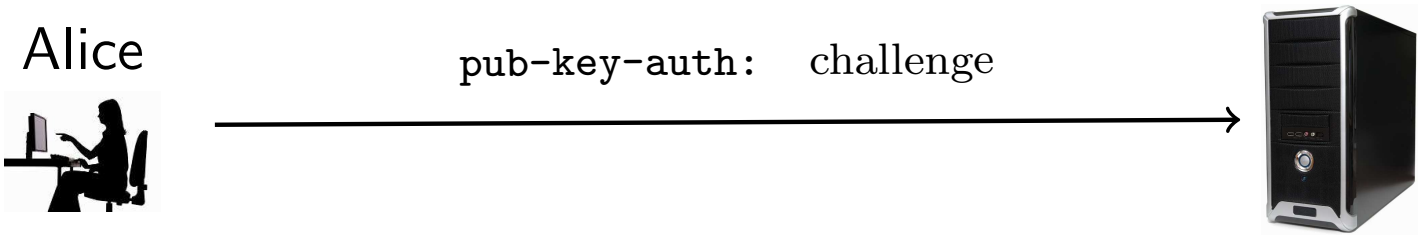
SSH Protocol

Alice

`pub-key-auth:` $E_{P_{\textbf{Alice}}}(\text{challenge})$

user	$P_{\textbf{user}}$
Alice	$P_{\textbf{Alice}}$
Bob	$P_{\textbf{Bob}}$

SSH Protocol



user	$P_{\textbf{user}}$
Alice	$P_{\textbf{Alice}}$
Bob	$P_{\textbf{Bob}}$

SSH Protocol

Alice

`Command prompt: "ssh>_"!`

user	$P_{\textbf{user}}$
Alice	$P_{\textbf{Alice}}$
Bob	$P_{\textbf{Bob}}$

Setting up SSH Tunneling

- Configure an SSH client to forward a local port to a port on a remote machine:

```
ssh -N -L 8888:192.168.111.222:80 bob@server
bob@server's password:.........
```

- 8888 is the local port.
- 192.168.111.222:80 is the IP address and the port on the remote machine.
- `-N` means not to execute any commands on the remote host.
- Start a browser, go to `http://localhost:8888`, this will redirect to 192.168.111.222:80.

SSH Tunneling...

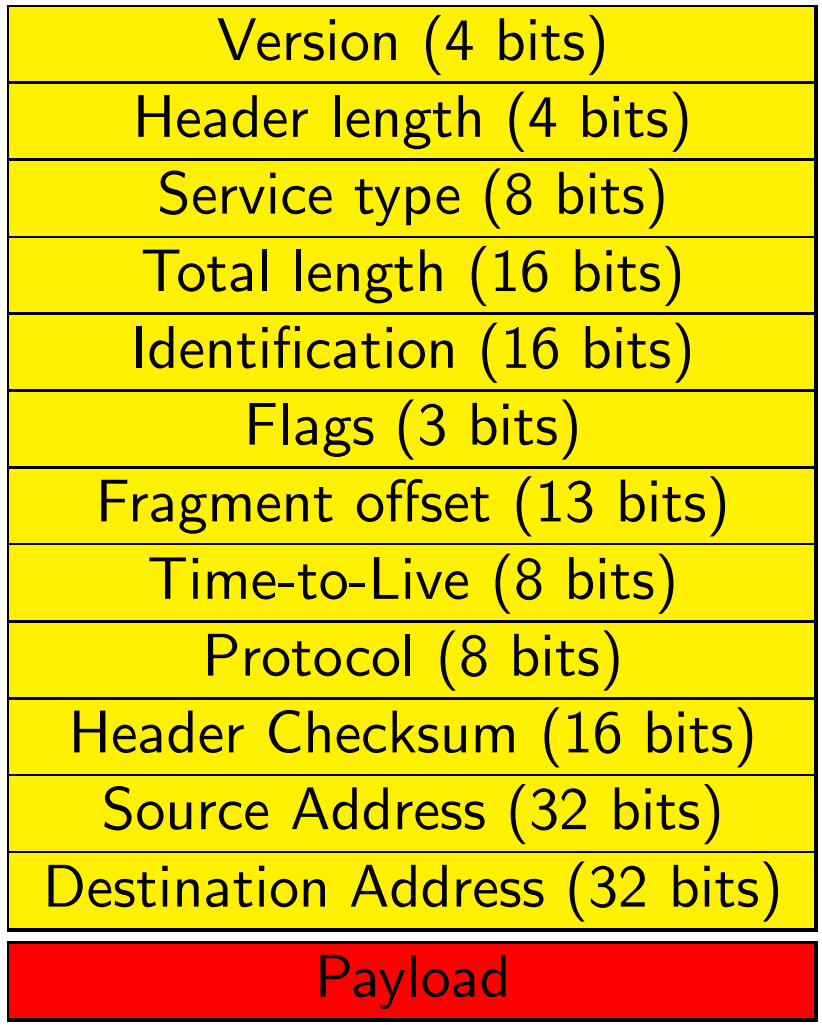
`-D [bind_address:]port:` Specifies a local "dynamic" application-level port forwarding. This works by allocating a socket to listen to port on the local side, optionally bound to the specified bind_address. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and the application protocol is then used to determine where to connect to from the remote machine.

`-L [bind_address:]port:host:hostport:` Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side. This works by allocating a socket to listen to port on the local side, optionally bound to the specified bind_address. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to host port hostport from the remote machine.

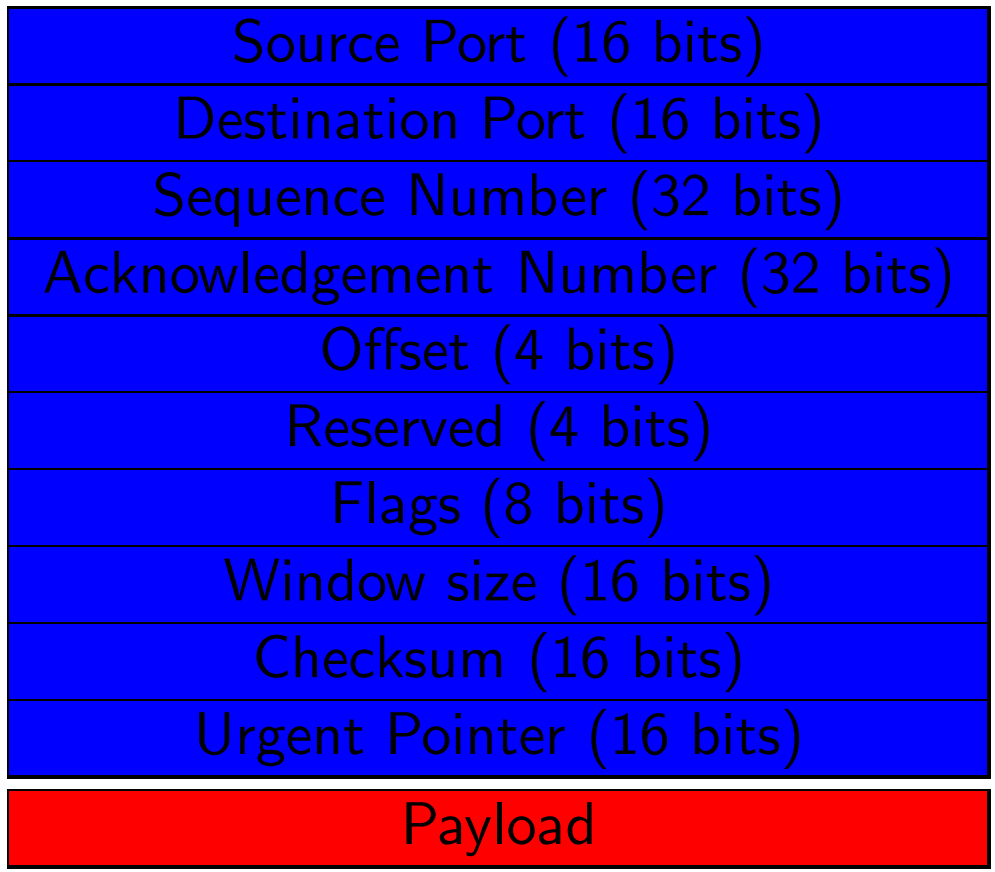
Internet Protocol Security (IPsec)

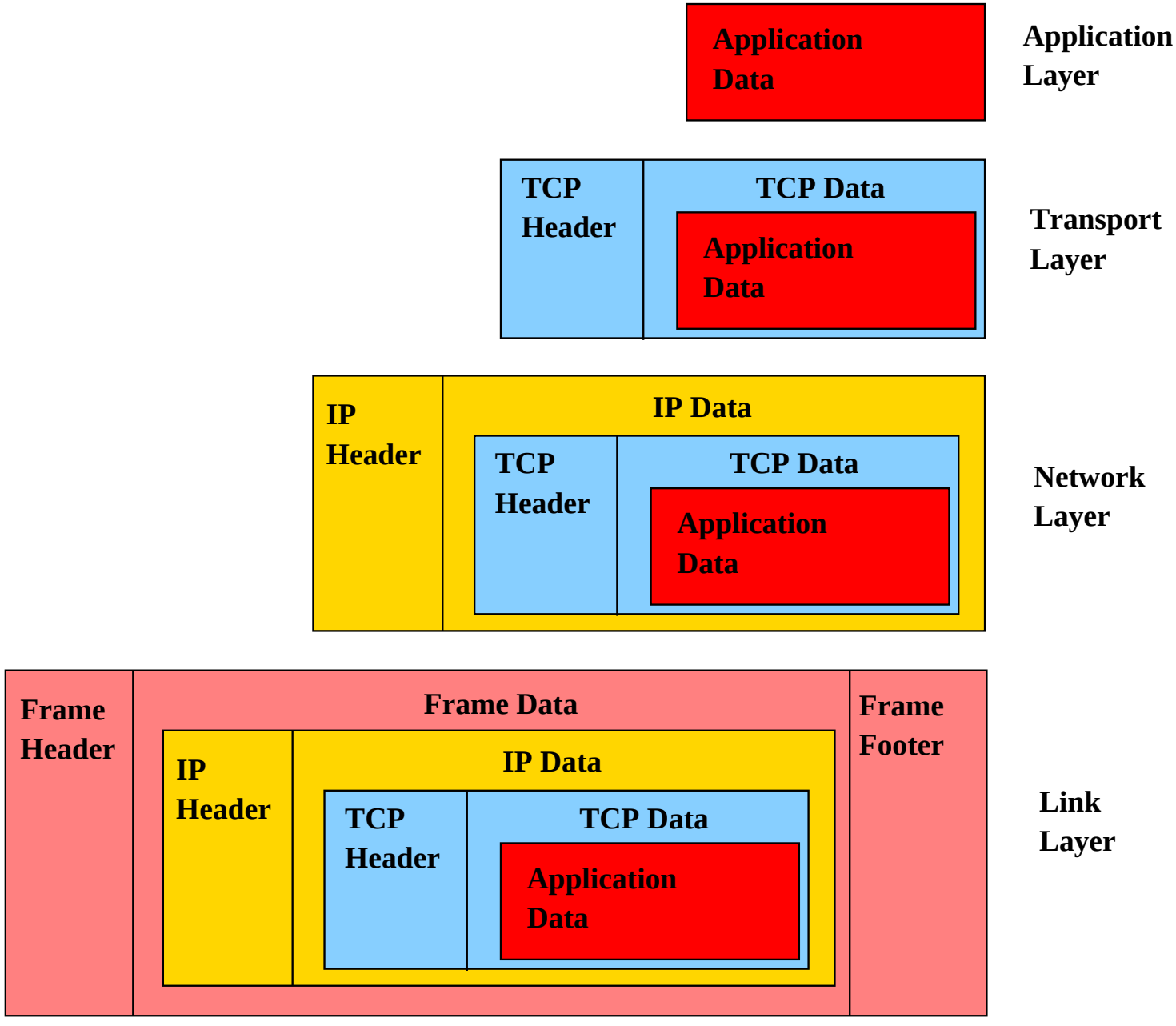
- Operates at the Internet (Network) Layer.
- Transparent to applications.
- Requires a modified IP stack.
- Network applications don't need to change.

IPv4 Packet Format

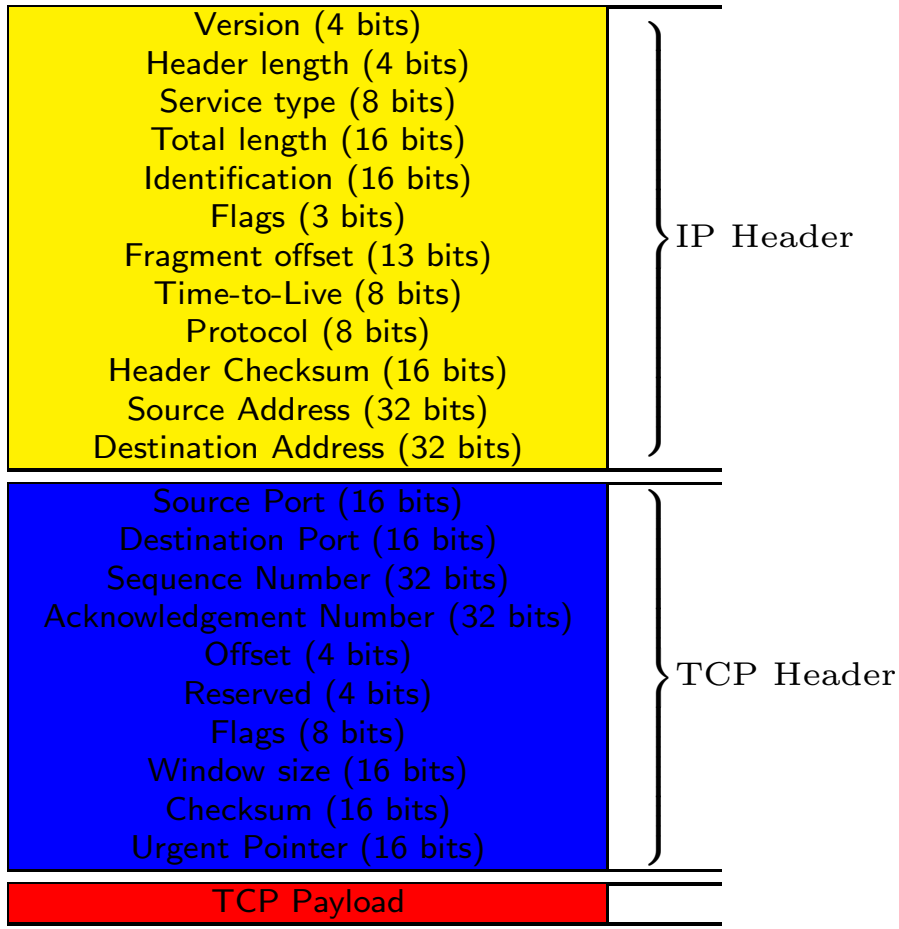


TCP Packet Format

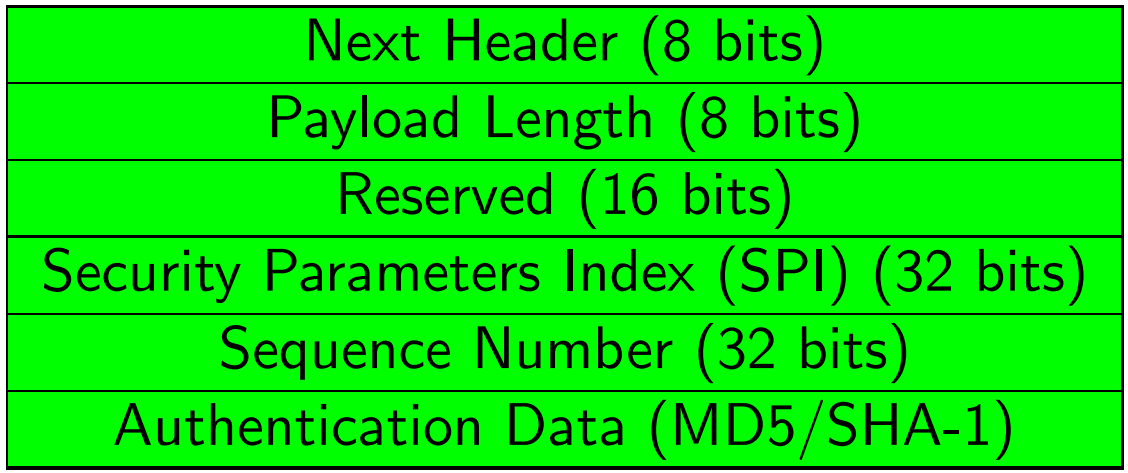



Application Data
Application Layer
TCP Header
TCP Data
Application Data
Transport Layer
IP Header
IP Data
TCP Header
TCP Data
Application Data
Network Layer
Frame Header
Frame Data
IP Header
IP Data
TCP Header
TCP Data
Application Data
Frame Footer
Link Layer

TPC Packet Encapsulated Within IPv4

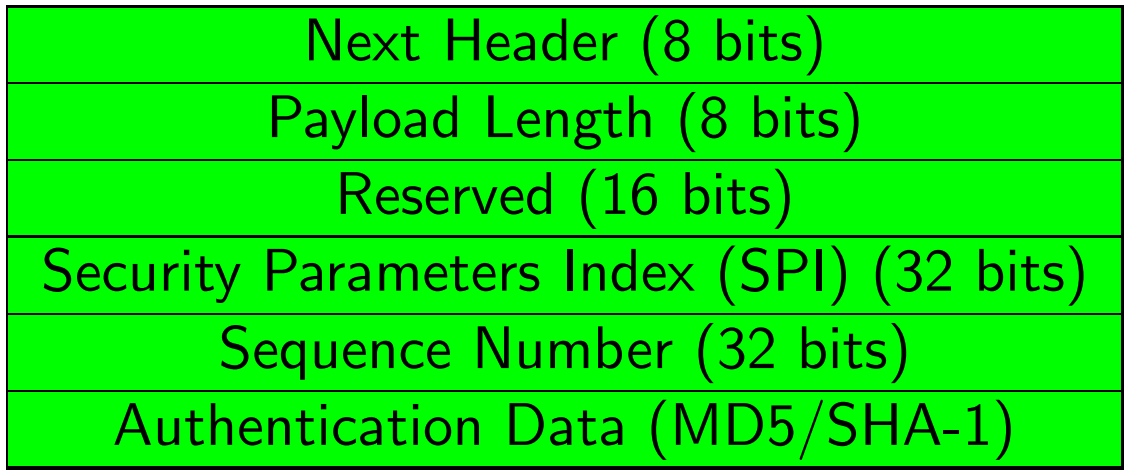


IPsec Authentication Header (AH)



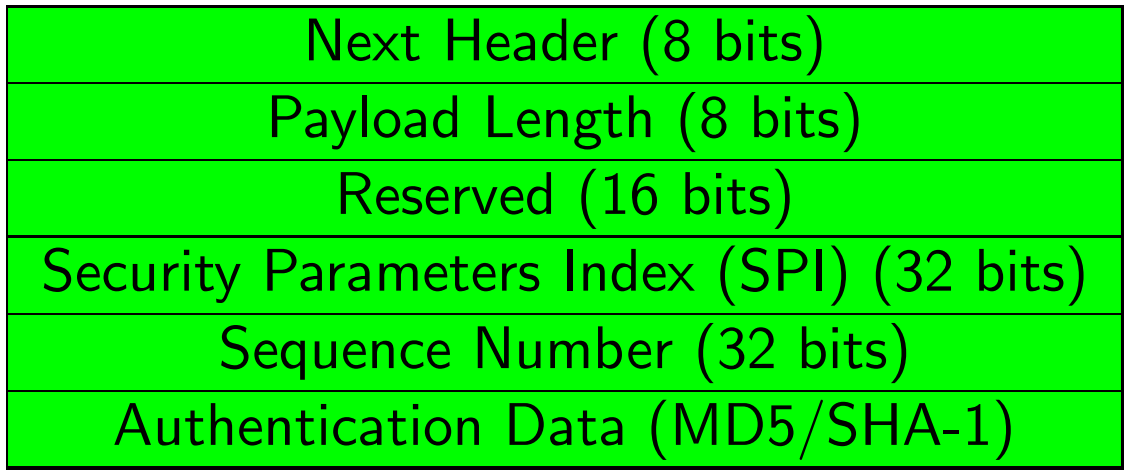
- The IPsec AH is used to authenticate the sender and ensure packet integrity.

IPsec Authentication Header (AH)



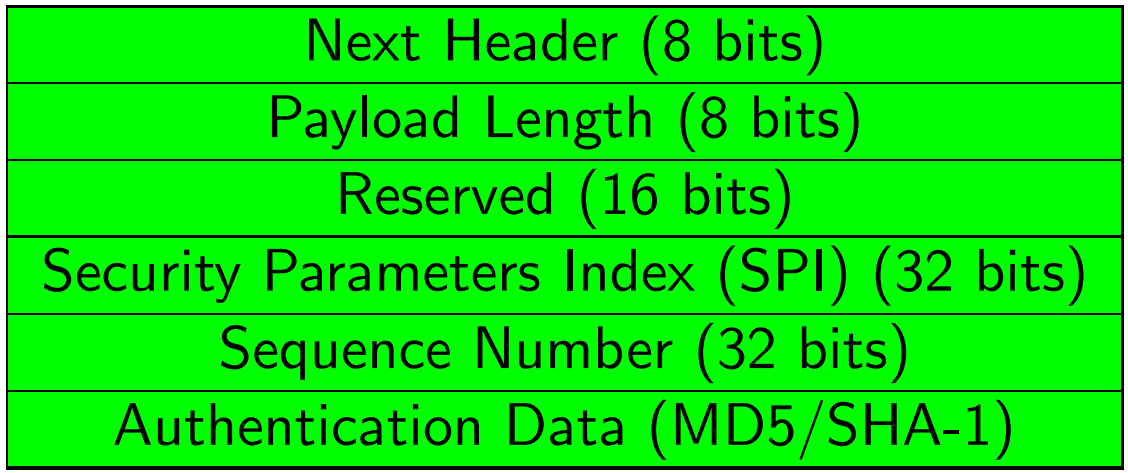
- The IPsec AH is used to authenticate the sender and ensure packet integrity.
- Security Parameters Index describes which cryptographic algorithms and keys to use.

IPsec Authentication Header (AH)



- The IPsec AH is used to authenticate the sender and ensure packet integrity.
- Security Parameters Index describes which cryptographic algorithms and keys to use.
- Randomly initialized Sequence Number is used to protect against replay attacks.

IPsec Authentication Header (AH)

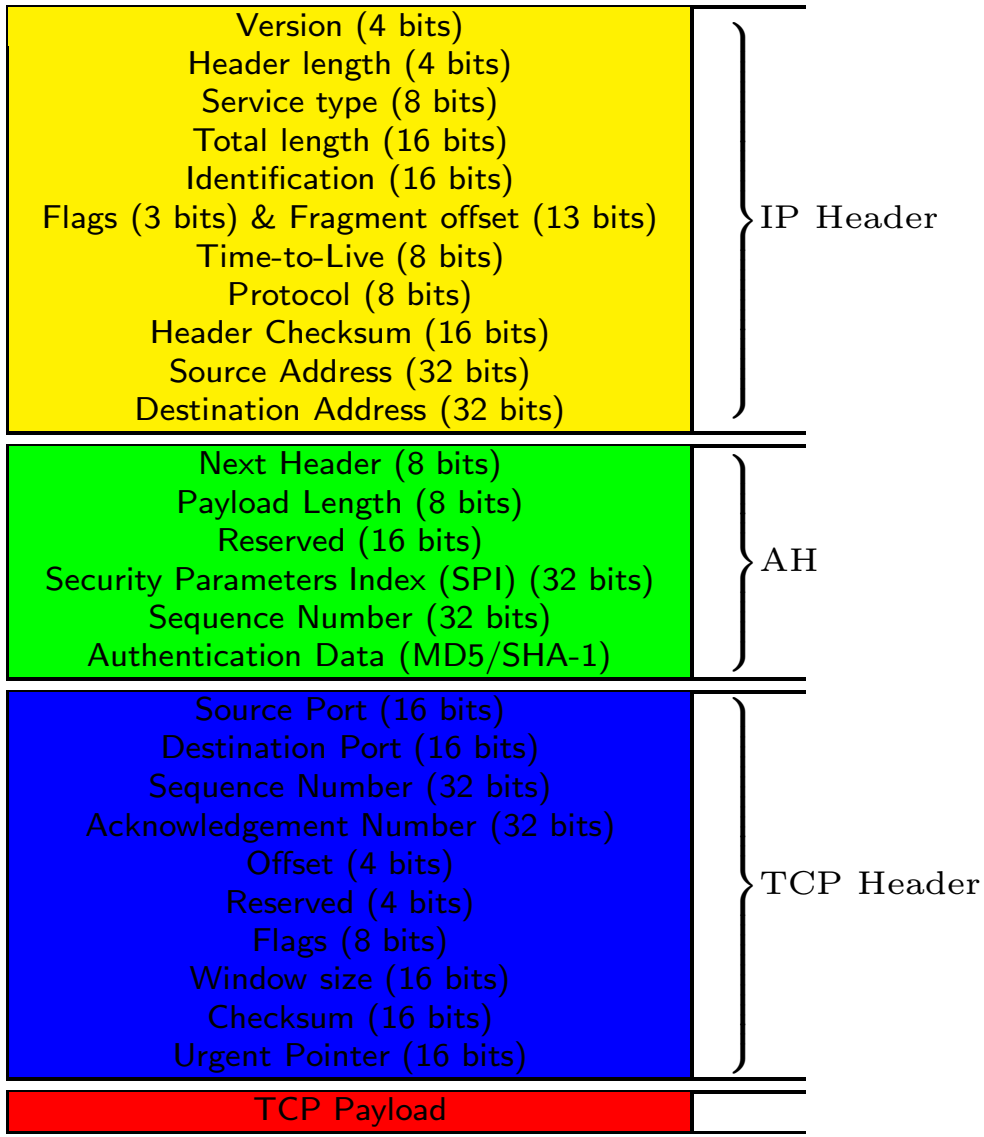


- The IPsec AH is used to authenticate the sender and ensure packet integrity.
- Security Parameters Index describes which cryptographic algorithms and keys to use.
- Randomly initialized Sequence Number is used to protect against replay attacks.
- Authentication Data contains an Integrity Check Value (ICV) to protect the integrity of the packet.

Transport Mode

- In Transport Mode we protect an end-to-end conversation between two hosts.
- We can authenticate and/or encrypt.
- We insert the AH header between the IP header and the TCP payload.
- Only the payload is encrypted and authenticated.

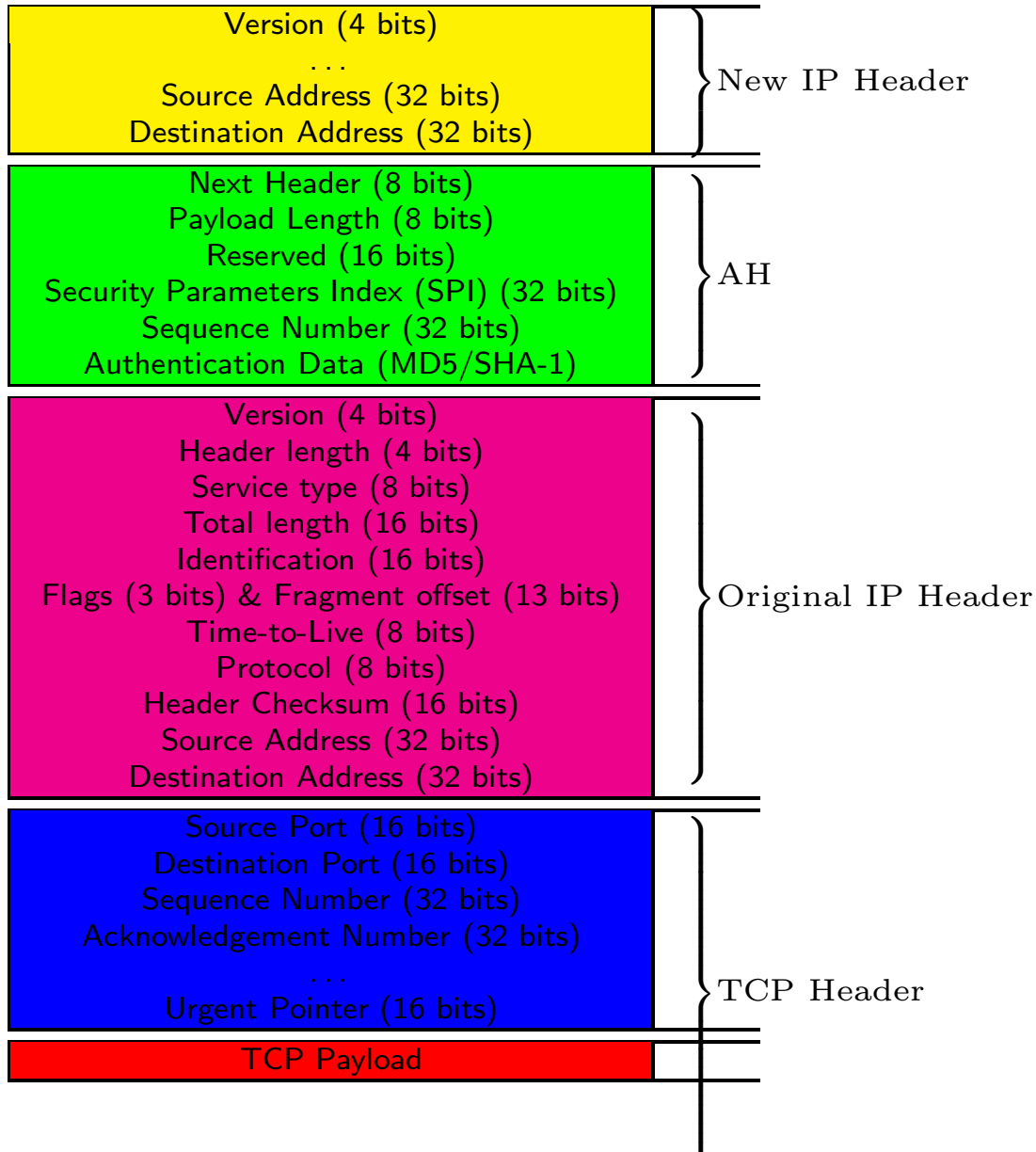
IPsec Packet in Transport Mode



Tunnel Mode

- In Tunnel Mode the entire IP packet is encapsulated inside another packet.
- Both original headers and payload are encrypted and authenticated.
- The source and destination addresses of the new packet can be different from those of the original packet. This allow us to form a tunnel.

IPsec Packet in Tunnel Mode



Security Associations (SAs)

- A set of Security Associations (SAs) tells, for a particular packet, which cryptographic keys and algorithms we should use to encrypt/decrypt/authenticate it.
- Each IPsec header has a Security Parameter Index (SPI), that indexes into a Security Association Database (SADB) and gives us the SAs.

Security Associations (SAs)...

- An SA defines the following parameters:
  1. Source and destination IP address
  2. IPsec protocol (AH or ESP)
  3. Cipher algorithm
  4. Secret key
  5. Security Parameter Index (SPI).

Security Association Datavase (SADB)

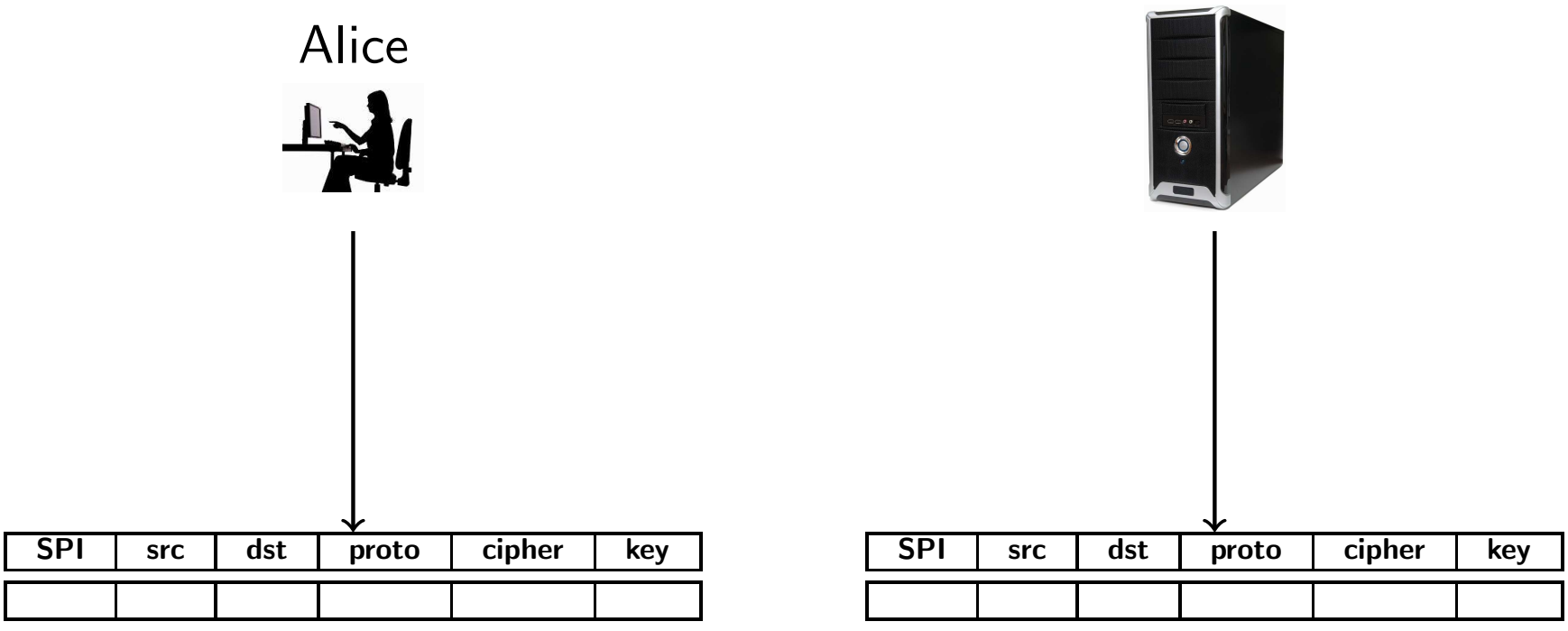
SPI	src	dest	protocol	cipher	key
1	176.11.1.2	192.10.2.3	AH	AES	0x4A5...
2	192.10.2.3	176.11.1.2	AH	3DES	0x56B...

- Note that you need one entry for each direction.
- Every packet includes the SPI so that the relevant keys and algorithms can be found.

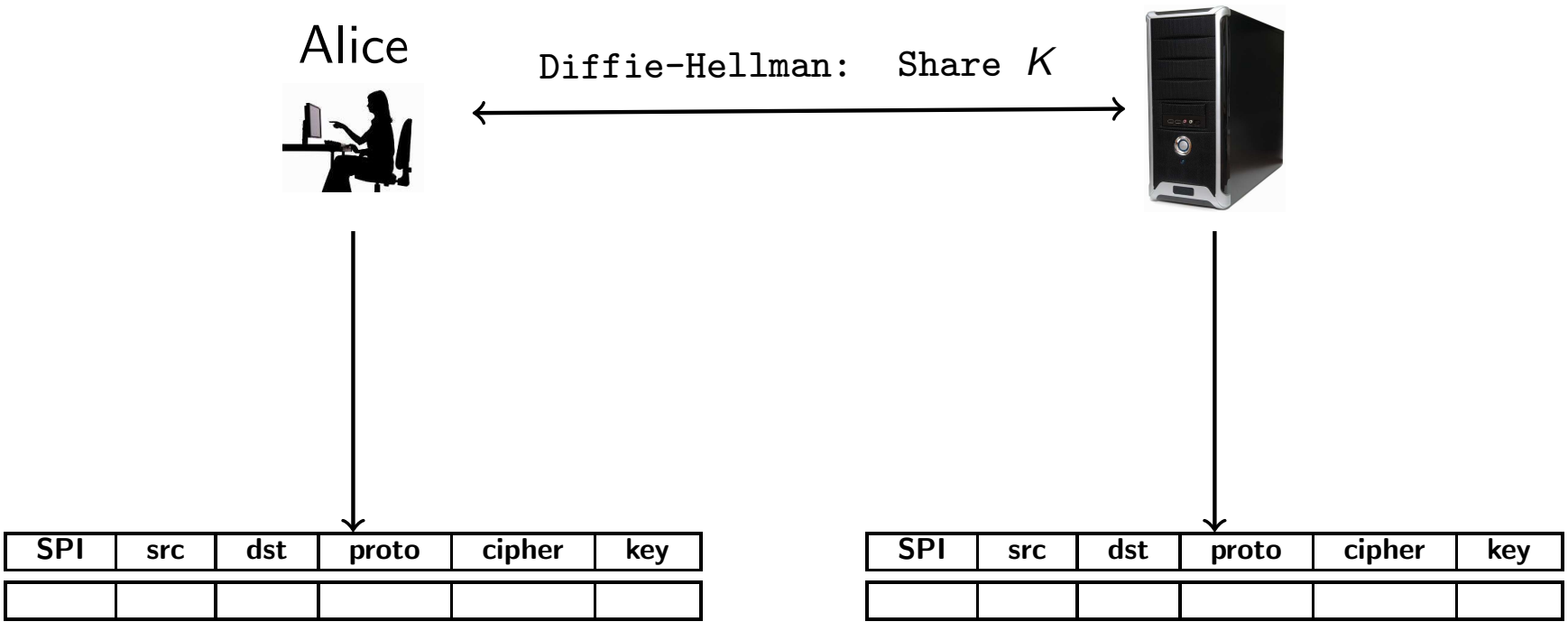
Internet Key Exchange (IKE)

- Before two processes can start to communicate, they must negotiate SAs.
- This is done using the Internet Key Exchange (IKE) protocol.
- IKE runs in two phases:
  1. Set up an inital SA to encrypt subsequent communication:
     - use DiffieHellman key exchange to generate a shared secret key K;
     - authenticate using pre-shared keys, signatures, or public-key encryption;
  2. Exchange SAs for the IPsec traffic (encrypt with K).

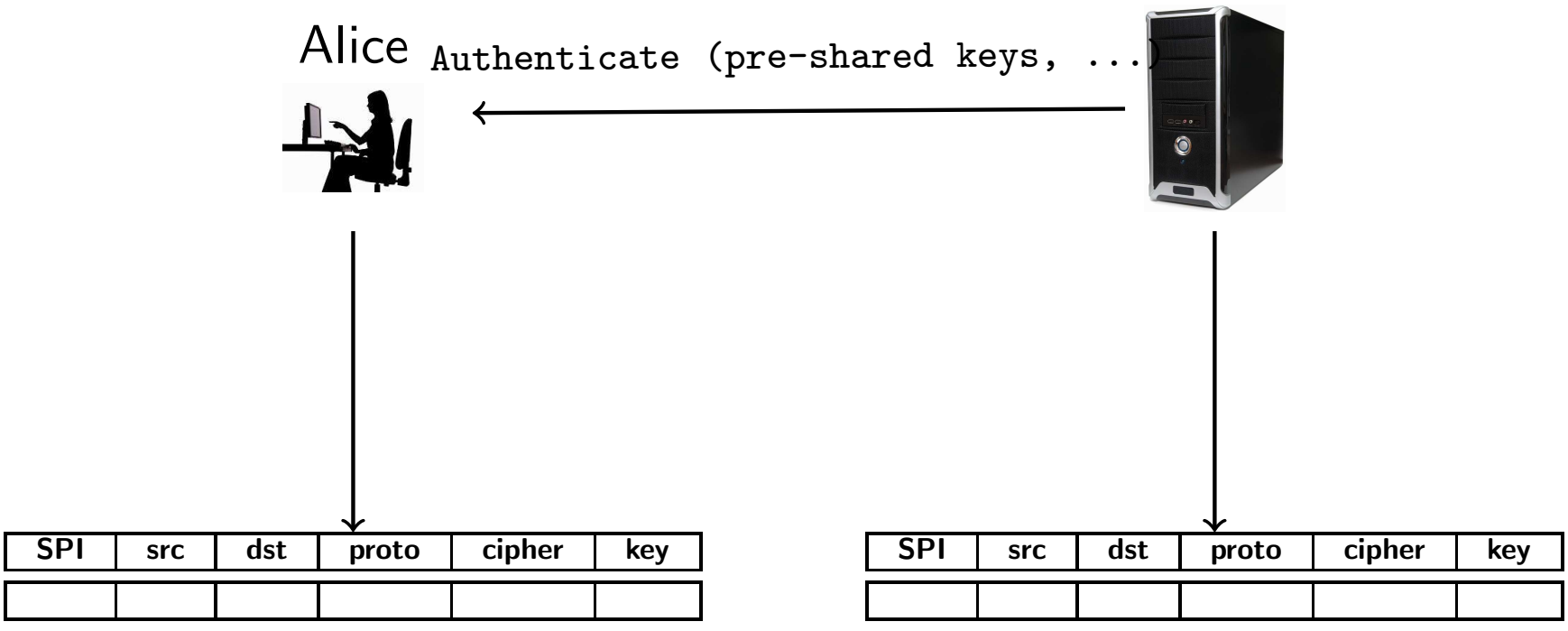
Internet Key Exchange (IKE)...



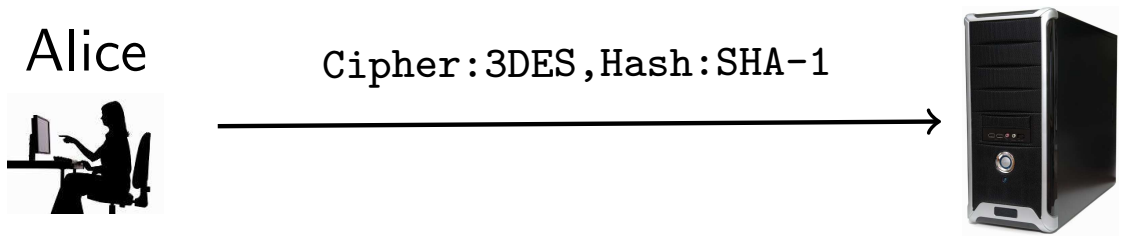
Internet Key Exchange (IKE)...



Internet Key Exchange (IKE)...



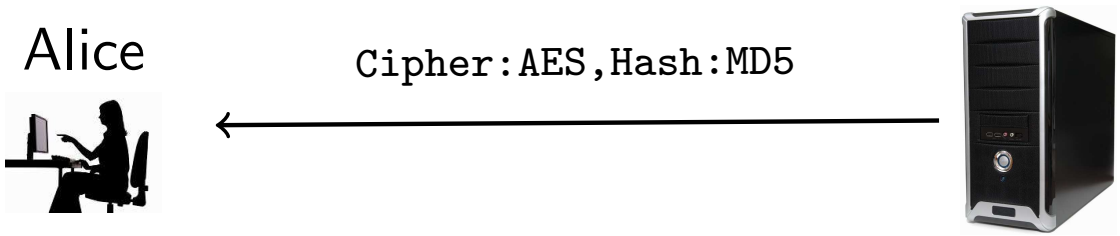
Internet Key Exchange (IKE)...



SPI	...	protocol	...
1	...	AES	...
2	...	3DES	...

SPI	...	protocol	...
1	...	3DES	...
2	...	AES	...

Internet Key Exchange (IKE)...



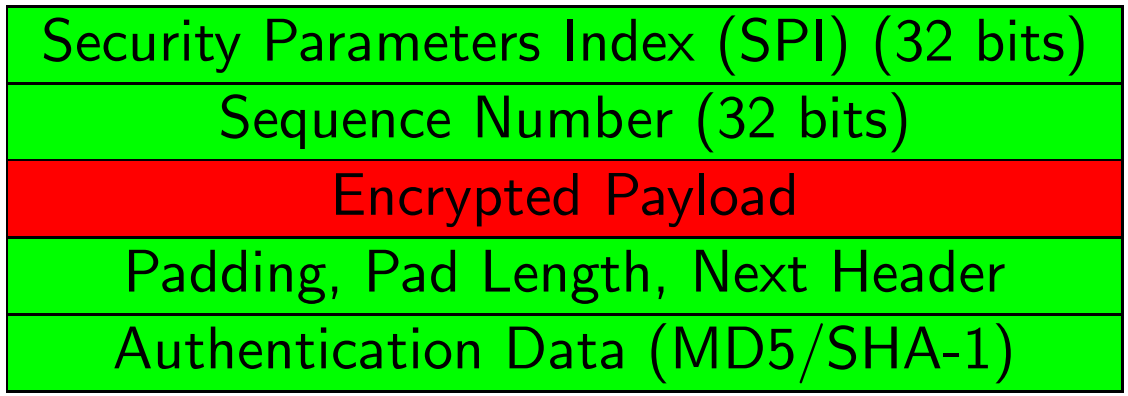
SPI	...	protocol	...
1	...	AES	...
2	...	3DES	...

SPI	...	protocol	...
1	...	3DES	...
2	...	AES	...

Integrity Check Value (ICV)

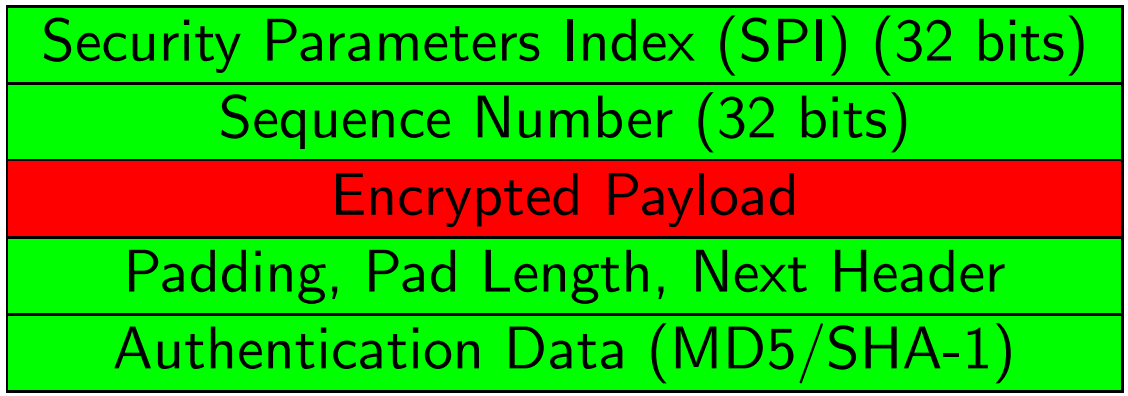
- The ICV is a hash over the entire packet, including IPsec header.
- A HMAC is used, incorporating a shared secret key:
  1. The hash provides tamper-detection.
  2. The key provides authentication.

IPsec Encapsulating Security Payload Header (ESP)



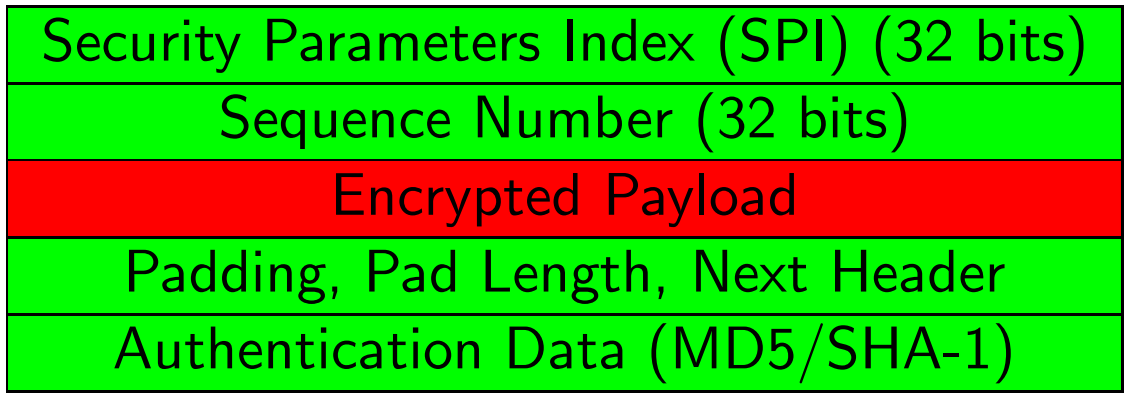
- IPsec defines two modes, AH and ESP.

IPsec Encapsulating Security Payload Header (ESP)



- IPsec defines two modes, AH and ESP.
- The AH mode only protects packet integrity, not confidentiality.

IPsec Encapsulating Security Payload Header (ESP)



- IPsec defines two modes, AH and ESP.
- The AH mode only protects packet integrity, not confidentiality.
- The Encapsulating Security Payload (ESP) mode also encrypts the payload.

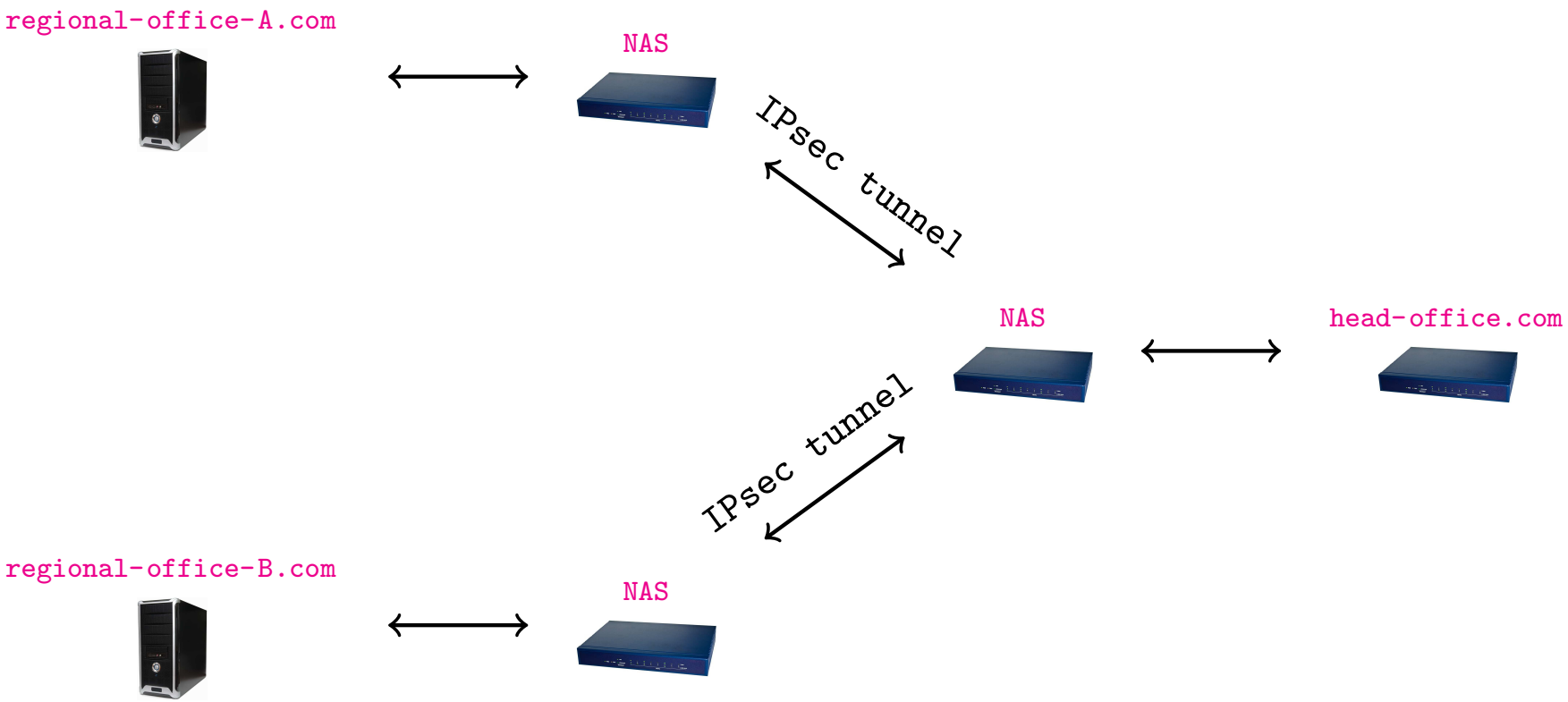
Virtual Private Network (VPN)

- Extend a private network (such as `arizona.edu`) over long physical distances.
- Olden days: lease a private line.
- Now-a-days: create a tunnel over the open Internet.
- To a VPN user, accessing a remote network is indistinguishable from being connected directly to the network.
- Two kinds:
  1. Remote Access VPN for roaming users
  2. Site-to-Site VPN for fixed networks.

Remote Access VPNs

- A Remote Access VPN connects a roaming user with a local network.
- The Network Access Server (NAS) is a VPN endpoint.

Site-to-Site VPNs



- A Site-to-Site VPN bridges two physically distant networks.
- Example: Connect the LANs at a company's head and regional offices.

Virtual Private Network — Problems

- Tunnels allow you to circumvent firewall policies:
  1. Exfiltration: a malicious insider can leak information from the company.
  2. Employees can surf unacceptable sites.

Outline

Intrusion Detection

- An Intrusion Detection System detects malicious activity on a network.
- Two components:
  1. IDS sensors: collect real-time data about the network,
  2. IDS manager: compiles reports from sensors.
- SNORT: http://www.youtube.com/watch?v=baxPhu1pA2M
- Research projekt ReMIND: http://www.youtube.com/watch?v=onHg5glDg_0

Sensor

Manager

Sensor

Sensor

Manager

Sensor

IDS Managers

- The manager
  1. compiles reports from sensors,
  2. decides if an intrusion has occurred,
  3. raises an alarm

IDS Sensors

- The sensors
  1. collect information from routers, servers, ...,
  2. writes records to an audit log.
- Audit log records:

  `[Subject,Object,Action,Error,Resource-Usage,Time-Stamp]`
- Examples:
  1. `[Alice,/etc/passwd,read,no-error,5ms,05:30]`
  2. `[128.72.100.99,196.200.11.2,HTTP,error-404,1ms,15:23]`

IDS Techniques — NIDS

- Network Intrusion Detection System (NIDS):
  - Where: edge of network;
  - Examines: traffic patterns, packet content;
  - Detects: malicious network behavior.
- Performs deep packet inspection in incoming and outgoing traffic.
- Rule based: compare traffic with a database of attack signatures.
- Statistical: caompare traffic against baseline network behavior.

IDS Techniques — PIDS

- Protocol-Based Intrusion Detection System (PIDS):
  - Where: network host;
  - Examines: specific protocol traffic;
  - Detects: malicius content.
- Examples:
  1. Web server runs a PID to detect and drop malicious HTTP requests.
  2. PIDS detects malformed SQL queries between web server and database server.

IDS Techniques — HIDS

- Host-Based Intrusion Detection System (HIDS):
  - Where: single machine;
  - Examines: system calls, resource usage, interprocess communication, system logs;
  - Detects: malicius users.
- A masquerader gains access using legitimate user's identity.
  - Detect unusual behavior using heuristic rules or statistical analysis.
- A misfeasor is a legitimate user behaving badly.
  - Detect violations of rules describing (un)authorized actions.
- A clandestine user deletes logs/audit files to cover up his actions.
  - Monitor (and log!) changes to log/audit files.

Attacks on IDSs

- DOS: deliberately trigger many intrusion alerts!
- Slow attacks!
- Learn and mimic normal behavior!

Port Scanning

- Port Scanning looks for open ports on a server.
- Information collected:
  - Open/Closed/Blocked connection?
  - Server operating system?
  - What service is running on a port?
- Can be used by
  - good guys (is my network secure?)
  - bad guys (what vulnerable services can I exploit?)

Port Scanning — `nmap`

Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

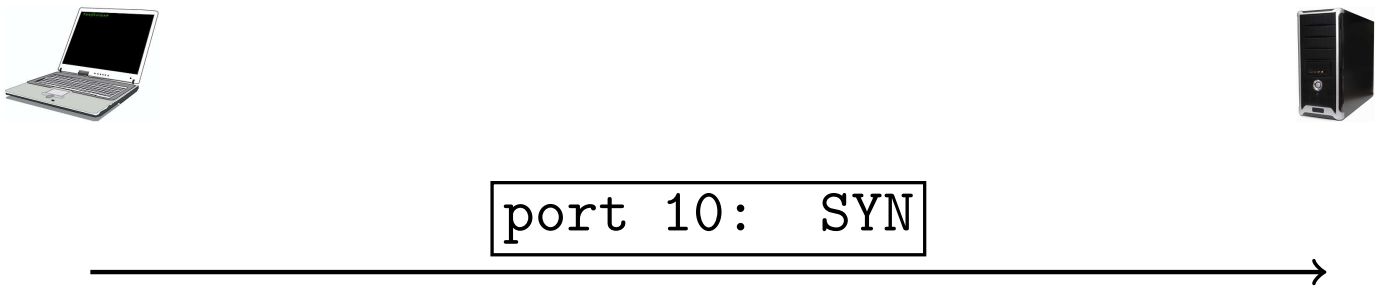
Port Scanning — nmapldots

- nmap -A -T4 scanme.nmap.org

SYN Scans

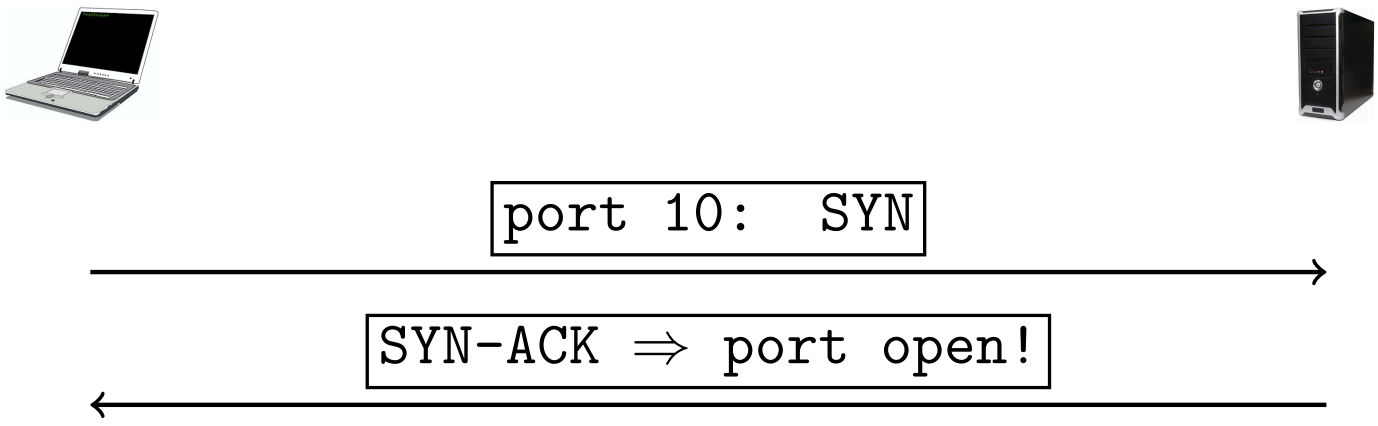
1. Send a TCP `SYN` packet to each port on the target.
2. Receive a `SYN-ACK` packet in return (the port is open!), close with `RET`.
3. Otherwise, try next port.

SYN Scans



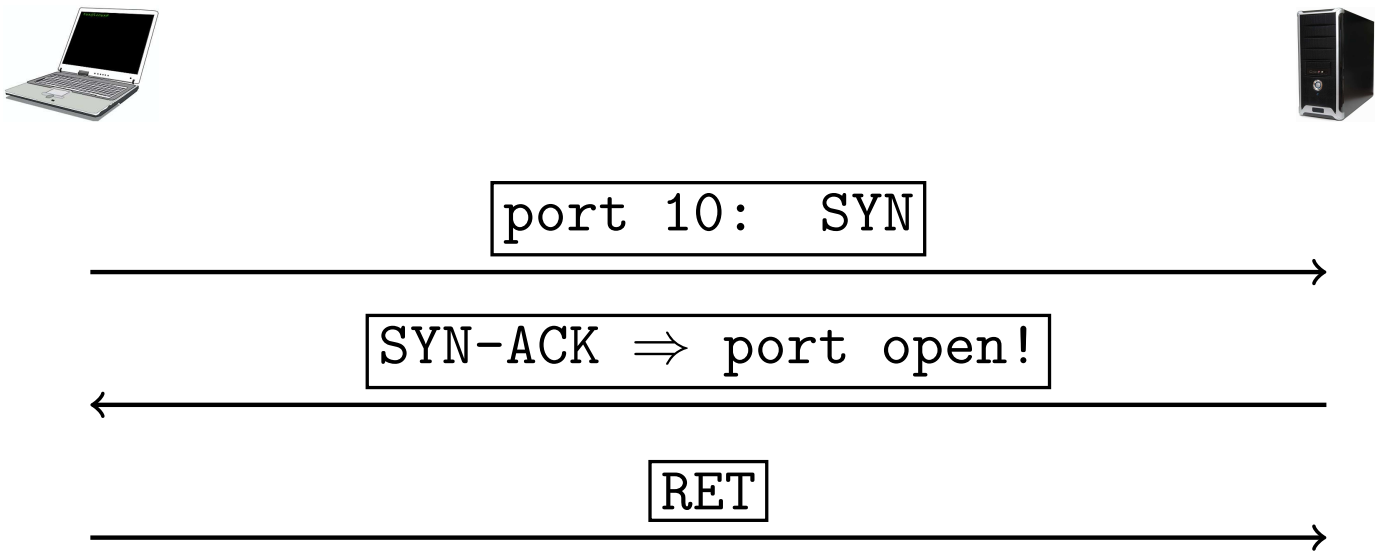
1. Send a TCP `SYN` packet to each port on the target.
2. Receive a `SYN-ACK` packet in return (the port is open!), close with `RET`.
3. Otherwise, try next port.

SYN Scans



1. Send a TCP `SYN` packet to each port on the target.
2. Receive a `SYN-ACK` packet in return (the port is open!), close with `RET`.
3. Otherwise, try next port.

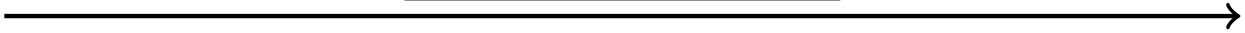
SYN Scans



1. Send a TCP `SYN` packet to each port on the target.
2. Receive a `SYN-ACK` packet in return (the port is open!), close with `RET`.
3. Otherwise, try next port.

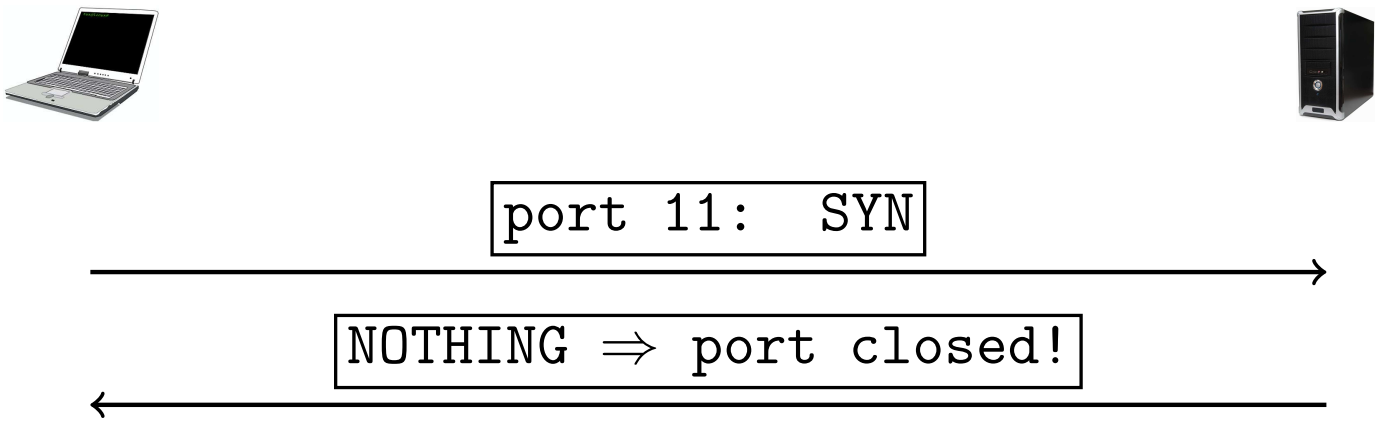
SYN Scans

`port 11: SYN`



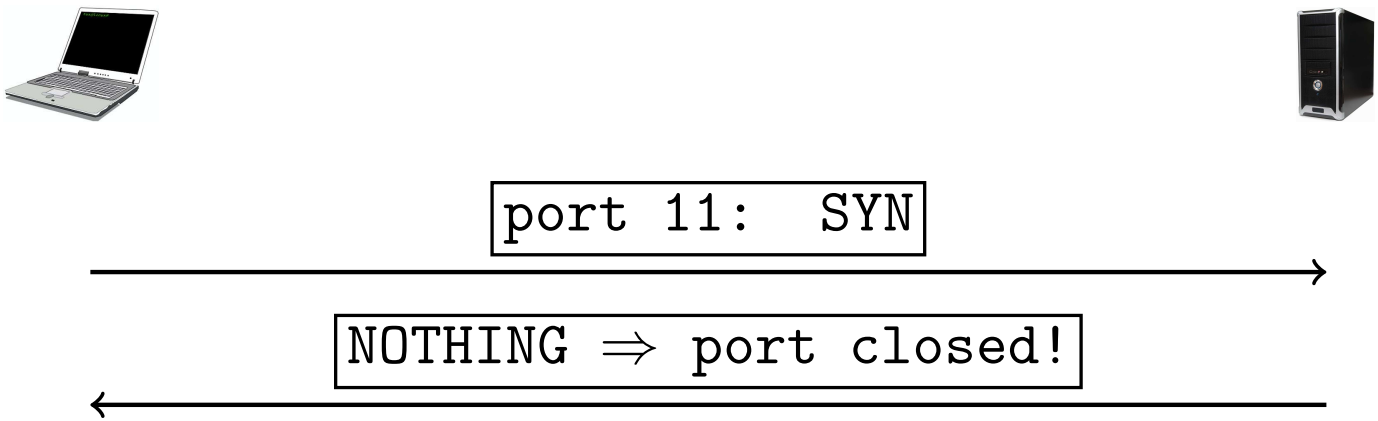
1. Send a TCP `SYN` packet to each port on the target.
2. Receive a `SYN-ACK` packet in return (the port is open!), close with `RET`.
3. Otherwise, try next port.

SYN Scans



1. Send a TCP `SYN` packet to each port on the target.
2. Receive a `SYN-ACK` packet in return (the port is open!), close with `RET`.
3. Otherwise, try next port.

SYN Scans



1. Send a TCP `SYN` packet to each port on the target.
2. Receive a `SYN-ACK` packet in return (the port is open!), close with `RET`.
3. Otherwise, try next port.

Outline

Readings and References

- Chapter 6 in *Introduction to Computer Security*, by Goodrich and Tamassia.