IF2230 Jaringan Komputer

Network Security

Robithoh Annur
Andreas Bara Timur
Monterico Andrian
Prof. Nana Rachmana

Security Vulnerabilities

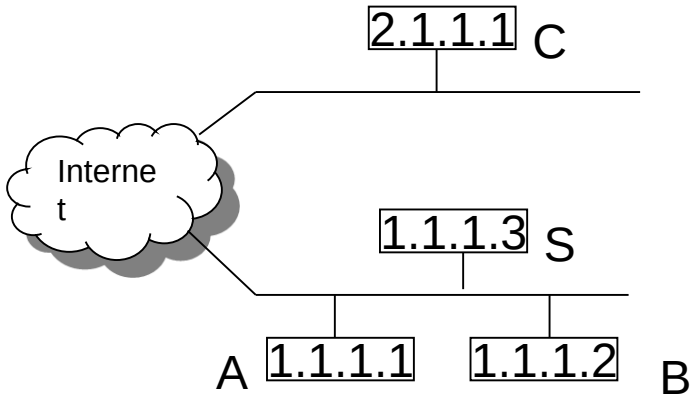
- At every layer in the protocol stack!
- Network-layer attacks
  - IP-level vulnerabilities
  - Routing attacks
- Transport-layer attacks
  - TCP vulnerabilities
- Application-layer attacks

IP-level vulnerabilities

- IP addresses are provided by the source
  - Spoofing attacks
- Using IP address for authentication
  - e.g., login with .rhosts
- Some “features” that have been exploited
  - Fragmentation
  - Broadcast for traffic amplification

Security Flaws in IP

- The IP addresses are filled in by the originating host
  - Address spoofing
- Using source address for authentication
  - r-utilities (rlogin, rsh, rhosts etc..)



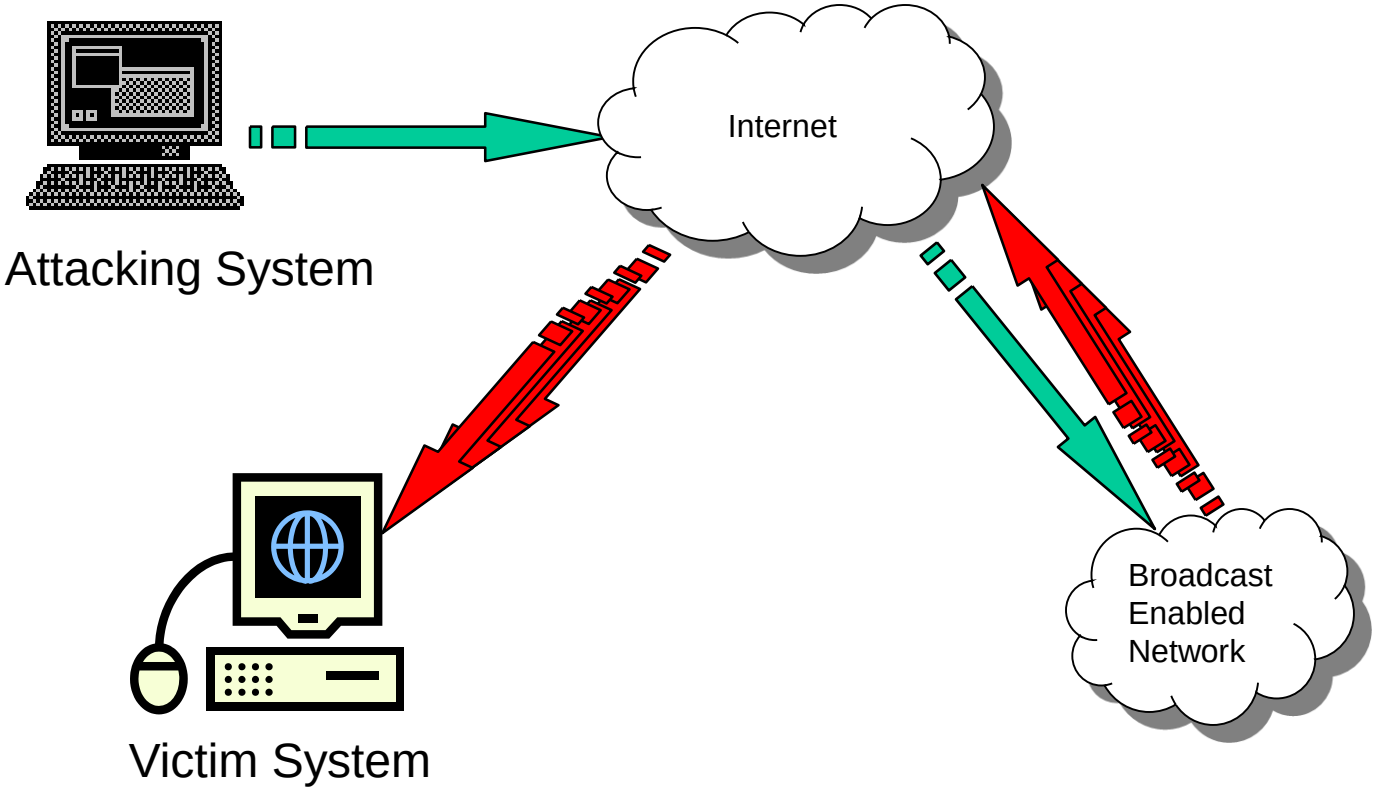
•Can A claim it is B to the server S?

•ARP Spoofing

•Can C claim it is B to the server S?

•Source Routing

Smurf Attack



ICMP Attacks

- No authentication
- ICMP redirect message
  - Can cause the host to switch gateways
  - Benefit of doing this?
    - Man in the middle attack, sniffing
- ICMP destination unreachable
  - Can cause the host to drop connection
- ICMP echo request/reply
- Many more…
  - http://www.sans.org/rr/whitepapers/threats/477.php

Routing attacks

- Divert traffic to malicious nodes
  - Black-hole
  - Eavesdropping
- How to implement routing attacks?
  - Distance-Vector:
  - Link-state:
- BGP vulnerabilities

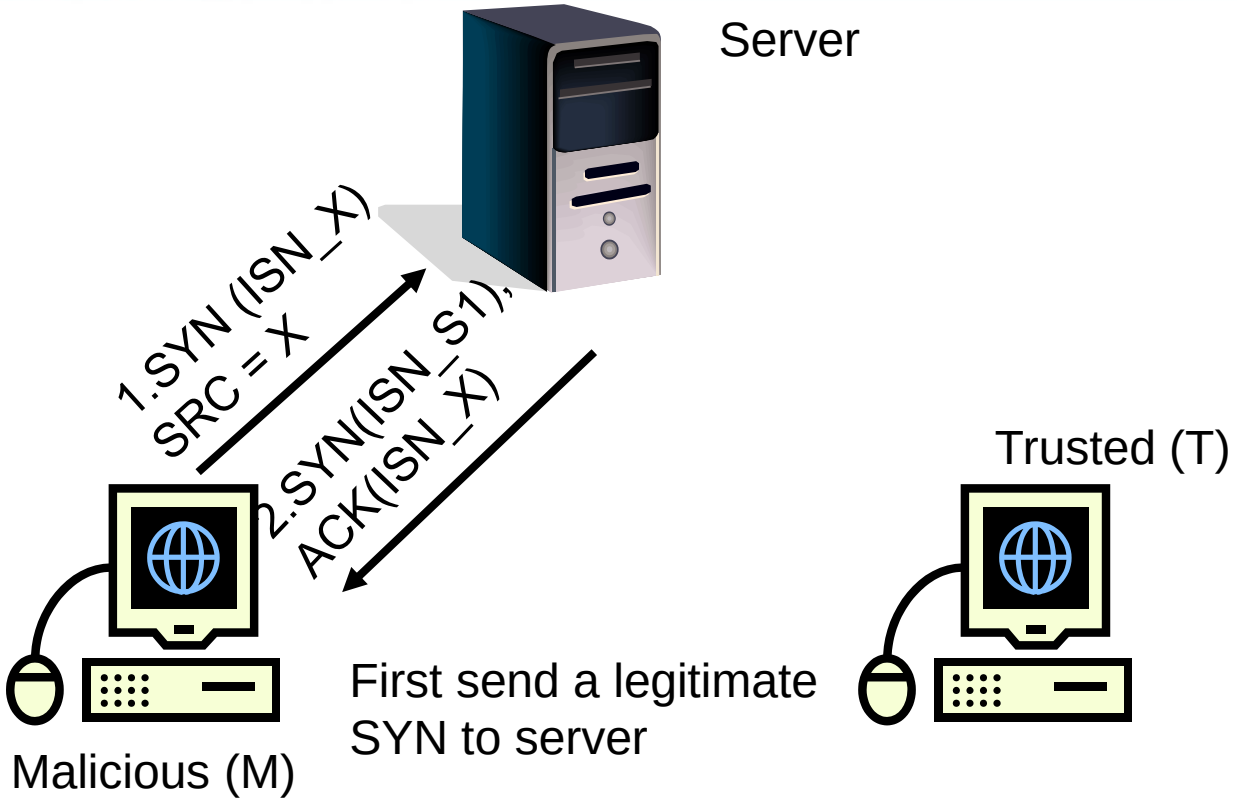
Routing attacks

- Divert traffic to malicious nodes
  - Black-hole
  - Eavesdropping
- How to implement routing attacks?
  - Distance-Vector: Announce low-cost routes
  - Link-state: Dropping links from topology
- BGP vulnerabilities
  - Prefix-hijacking
  - Path alteration

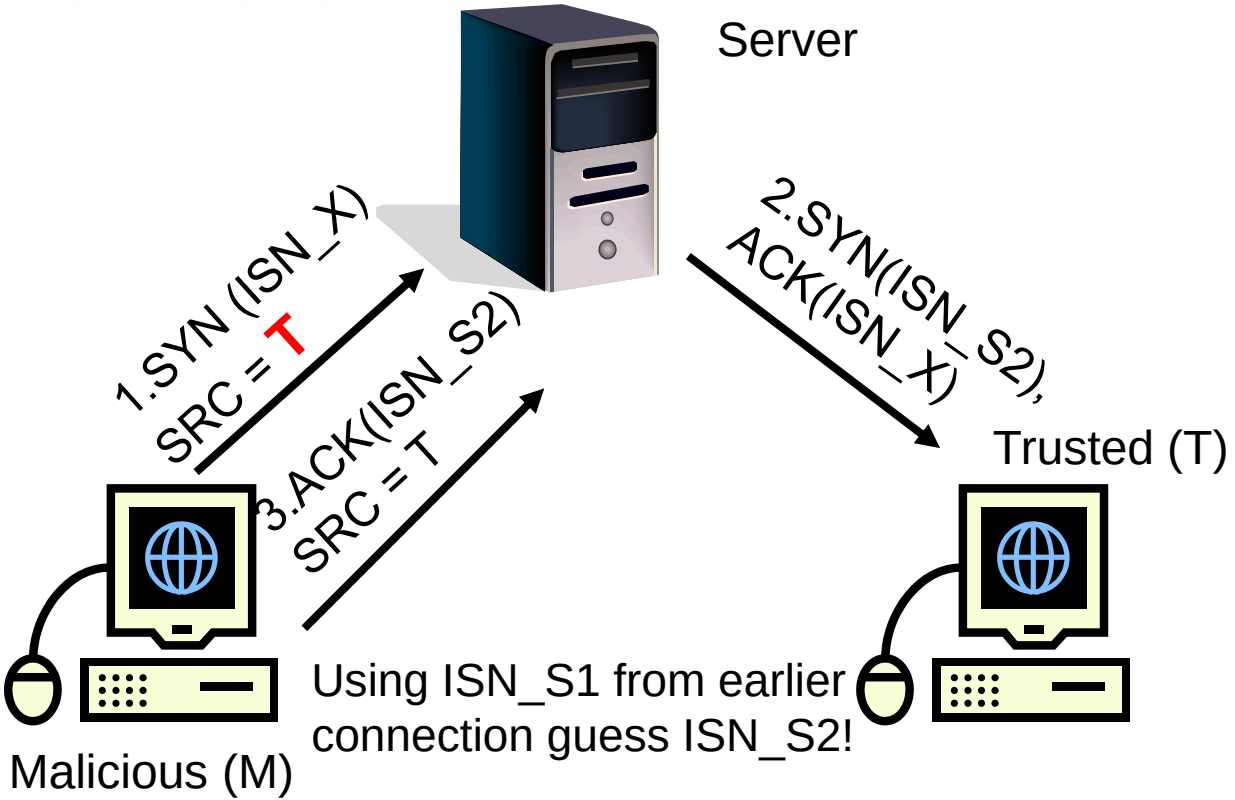
TCP-level attacks

- SYN-Floods
  - Implementations create state at servers before connection is fully established
- Session hijack
  - Pretend to be a trusted host
  - Sequence number guessing
- Session resets
  - Close a legitimate connection

Session Hijack



Session Hijack



TCP Layer Attacks

- TCP SYN Flooding
  - Exploit state allocated at server after initial SYN packet
  - Send a SYN and don’t reply with ACK
  - Server will wait for 511 seconds for ACK
  - Finite queue size for incomplete connections (1024)
  - Once the queue is full it doesn’t accept requests

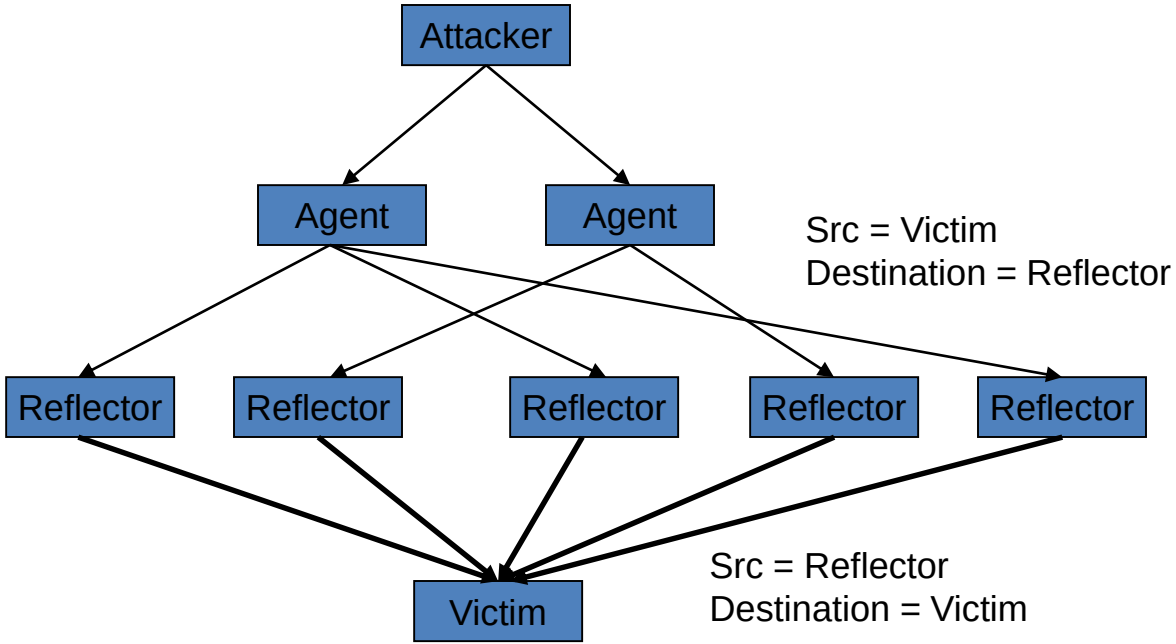
TCP Layer Attacks

- TCP Session Poisoning
  - Send RST packet
    - Will tear down connection
  - Do you have to guess the exact sequence number?
    - Anywhere in window is fine
    - For 64k window it takes 64k packets to reset
    - About 15 seconds for a T1

Denial of Service

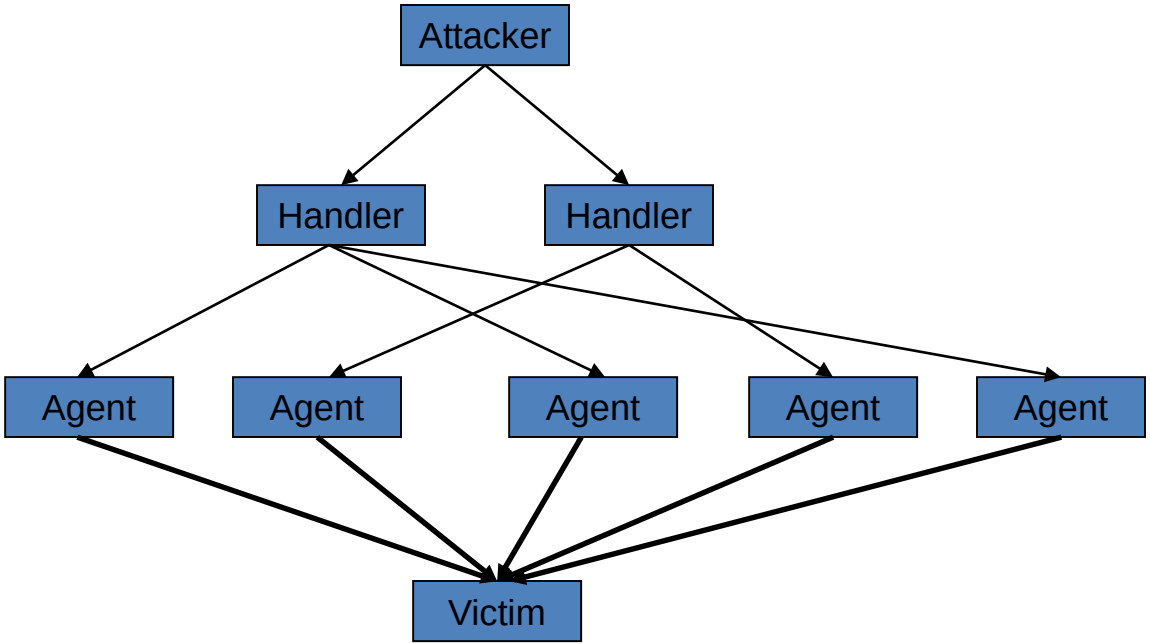
- Make a service unusable/unavailable
- Disrupt service by taking down hosts
  - E.g., ping-of-death
- Consume host-level resources
  - E.g., SYN-floods
- Consume network resources
  - E.g., UDP/ICMP floods

Reflector Attack



Unsolicited traffic at victim from legitimate hosts

Distributed DoS



Distributed DoS

- Handlers are usually high volume servers
  - Easy to hide the attack packets
- Agents are usually home users with DSL/Cable
  - Already infected and the agent installed
- Very difficult to track down the attacker
  - Multiple levels of indirection!
- Aside: How to distinguish DDos from flash crowd?

Worm Overview

- Self-propagate through network
- Typical Steps in worm propagation
  - Probe host for vulnerable software
  - Exploit the vulnerability (e.g., buffer overflow)
    - Attacker gains privileges of the vulnerable program
  - Launch copy on compromised host
- Spread at exponential rate
  - 10M hosts in < 5 minutes
  - Hard to deal with manual intervention

Scanning Techniques

- Random
- Local subnet
- Routing Worm
- Hitlist
- Topological

Random Scanning

- 32-bit randomly generated IP address
  - E.g., Slammer and Code Red I
  - What about IPv6?

- Hits black-holed IP space frequently
  - Only 28.6% of IP space is allocated
  - Detect worms by monitoring unused addresses
    - Honeypots/Honeynet

Subnet Scanning

- Generate last 1, 2, or 3 bytes of IP address randomly
- Code Red II and Blaster
- Some scans must be completely random to infect whole internet

Some proposals for countermeasures

- Better software safeguards
  - Static analysis and array bounds checking (lint/e-fence)
  - Safe versions of library calls
    - gets(buf) -> fgets(buf, size, ...)
    - sprintf(buf, ...) -> snprintf(buf, size, ...)
- Host-diversity
  - Avoid same exploit on multiple machines
- Network-level: IP address space randomization
- Host-level solutions
  - E.g., Memory randomization, Stack guard
- Rate-limiting: Contain the rate of spread
- Content-based filtering: signatures in packet payloads

Countermeasure Overview

- High level basic approaches
  - Prevention
  - Detection
  - Resilience
- Requirements
  - Security: soundness / completeness (false positive / negative
  - Overhead
  - Usability

Design questions ..

- Why is it so easy to send unwanted traffic?
  - Worm, DDoS, virus, spam, phishing etc
- Where to place functionality for stopping unwanted traffic?
  - Edge vs. Core
  - Routers vs. Middleboxes
- Redesign Internet architecture to detect and prevent unwanted traffic?

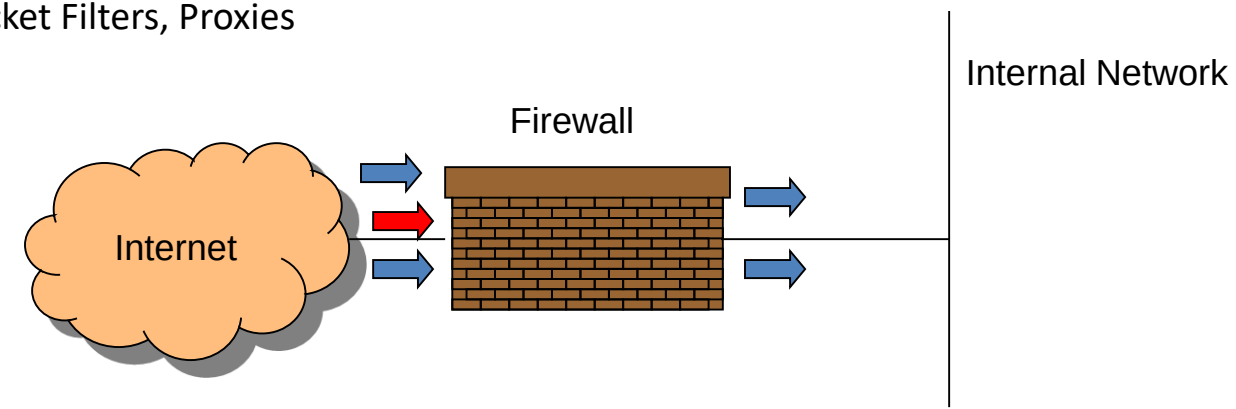
Firewalls

- Block/filter/modify traffic at network-level
  - Limit access to the network
  - Installed at perimeter of the network

- Why network-level?
  - Vulnerabilities on many hosts in network
  - Users don’t keep systems up to date
  - Lots of patches to keep track of
  - Zero-day exploits

Firewalls (contd…)

- Firewall inspects traffic through it
- Allows traffic specified in the policy
- Drops everything else
- Two Types
  - Packet Filters, Proxies



Packet Filters

- Selectively passes packets from one network interface to another
- Usually done within a router between external and internal network
- What/How to filter?
  - Packet Header Fields
    - IP source and destination addresses
    - Application port numbers
    - ICMP message types/ Protocol options etc.
  - Packet contents (payloads)

Packet Filters: Possible Actions

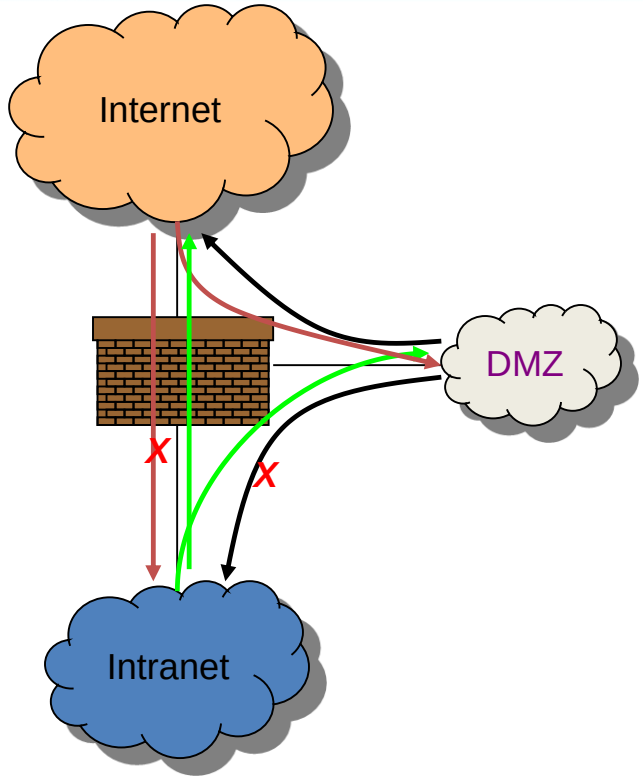
- Allow the packet to go through
- Drop the packet (Notify Sender/Drop Silently)
- Alter the packet (NAT?)
- Log information about the packet

Some examples

- Block all packets from outside except for SMTP servers
- Block all traffic to/from a list of domains
- Ingress filtering
  - Drop pkt from outside with addresses inside the network
- Egress filtering
  - Drop pkt from inside with addresses outside the network

Typical Firewall Configuration

- Internal hosts can access DMZ and Internet
- External hosts can access DMZ only, not Intranet
- DMZ hosts can access Internet only
- Advantages?
  - If a service gets compromised in DMZ it cannot affect internal hosts



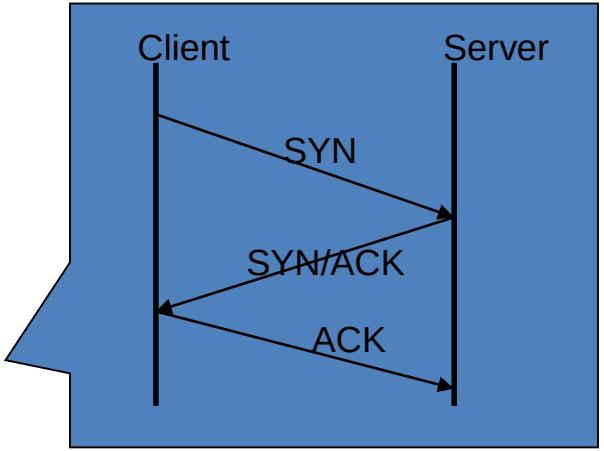
Firewall implementation

- Stateless packet filtering firewall
- Rule → (Condition, Action)
- Rules are processed in top-down order
  - If a condition satisfied – action is taken

Sample Firewall Rule

Allow SSH from external hosts to internal hosts

- Two rules
  - Inbound and outbound
- How to know a packet is for SSH?
  - Inbound: src-port>1023, dst-port=22
  - Outbound: src-port=22, dst-port>1023
  - Protocol=TCP
- Ack Set?
- Problems?



Rule	Dir	Src Addr	Src Port	Dst Addr	Dst Port	Proto	Ack Set?	Action
SSH-1	In	Ext	> 1023	Int	22	TCP	Any	Allow
SSH-2	Out	Int	22	Ext	> 1023	TCP	Yes	Alow

Default Firewall Rules

- Egress Filtering
  - Outbound traffic from external address → Drop
  - Benefits?
- Ingress Filtering
  - Inbound Traffic from internal address → Drop
  - Benefits?
- Default Deny
  - Why?

Rule	Dir	Src Addr	Src Port	Dst Addr	Dst Port	Proto	Ack Set?	Action
Egress	Out	Ext	Any	Ext	Any	Any	Any	Deny
Ingress	In	Int	Any	Int	Any	Any	Any	Deny
Default	Any	Any	Any	Any	Any	Any	Any	Deny

Packet Filters

- Advantages
  - Transparent to application/user
  - Simple packet filters can be efficient

- Disadvantages
  - Usually fail open
  - Very hard to configure the rules
  - May only have coarse-grained information?
    - Does port 22 always mean SSH?
    - Who is the user accessing the SSH?

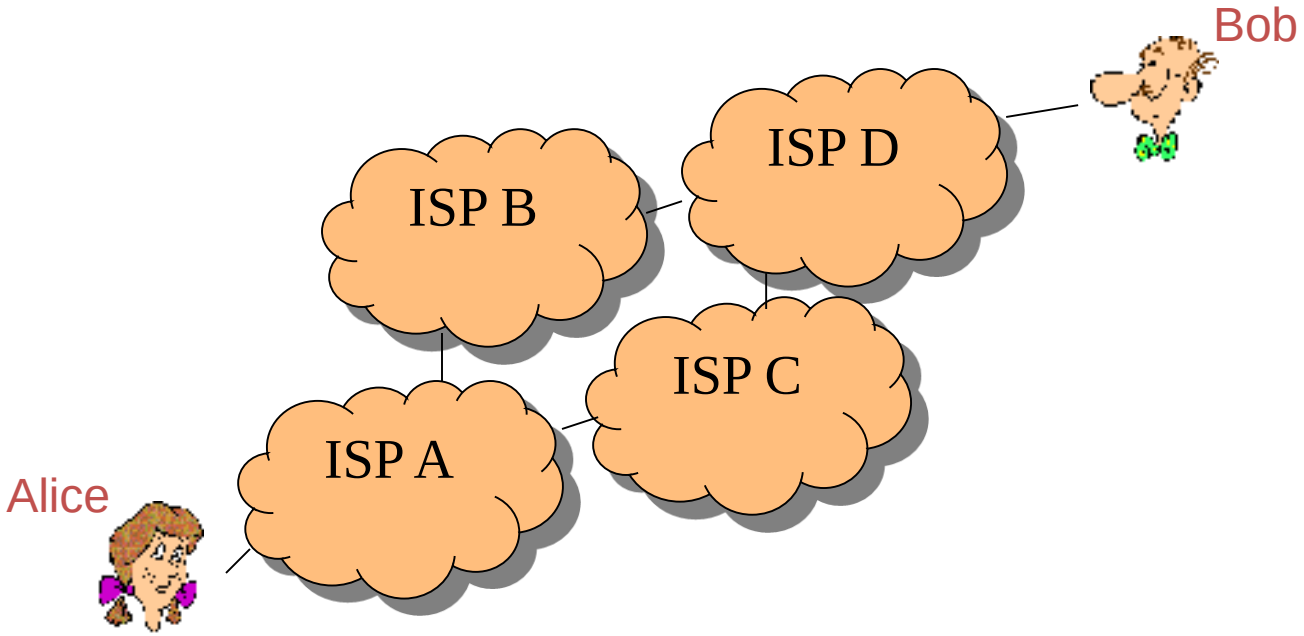
Alternatives

- Stateful packet filters
  - Keep the connection states
  - Easier to specify rules
  - Problems?
    - State explosion
    - State for UDP/ICMP?
- Proxy Firewalls
  - Two connections instead of one
  - Either at transport level
    - SOCKS proxy
  - Or at application level
    - HTTP proxy

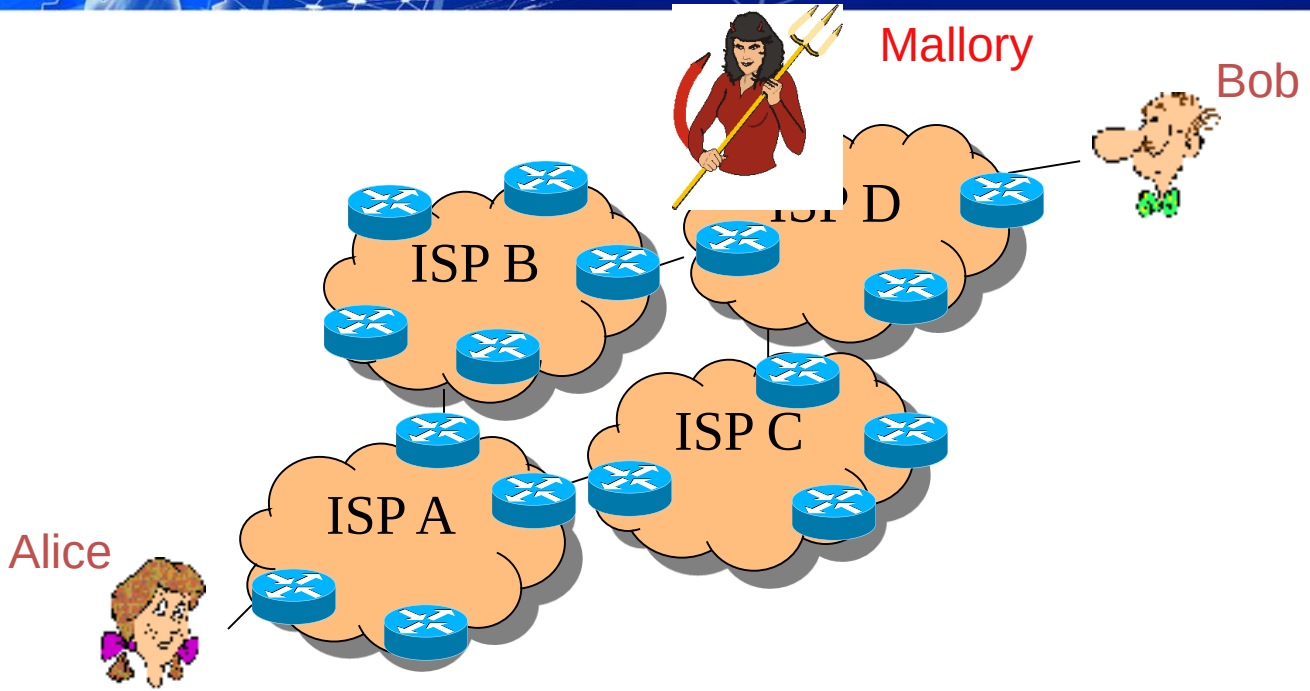
Intrusion Detection Systems

- Firewalls allow traffic only to legitimate hosts and services
- Traffic to the legitimate hosts/services can have attacks
- Solution?
  - Intrusion Detection Systems
  - Monitor data and behavior
  - Report when identify attacks

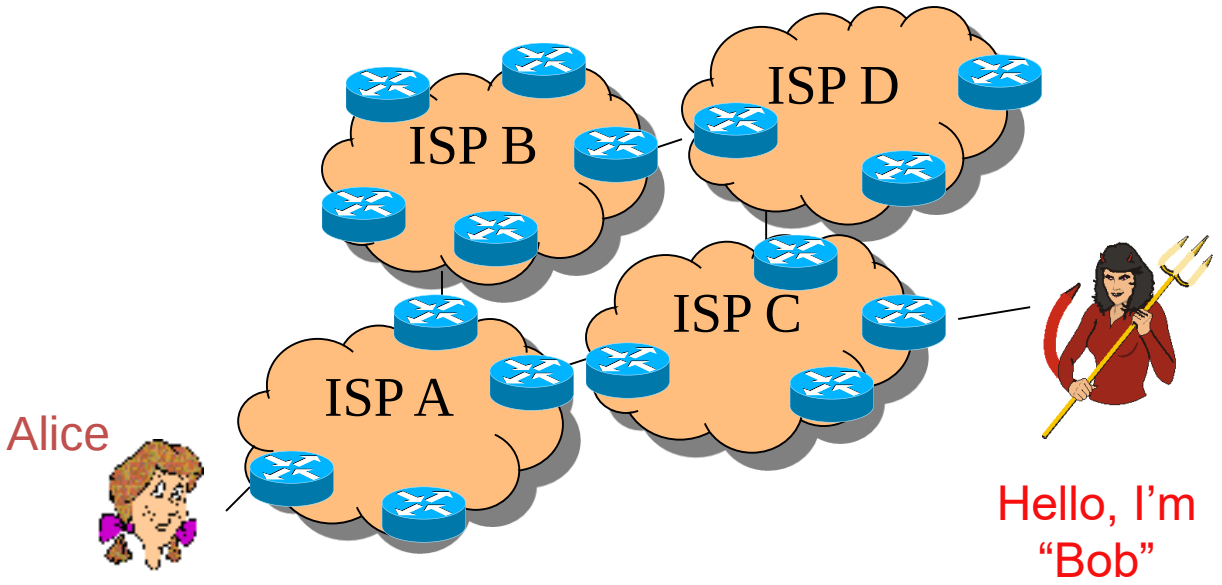
Secure Communication with an Untrusted Infrastructure



Secure Communication with an Untrusted Infrastructure



Secure Communication with an Untrusted Infrastructure



What do we need for a secure comm channel?

- Authentication (Who am I talking to?)
- Confidentiality (Is my data hidden?)
- Integrity (Has my data been modified?)
- Availability (Can I reach the destination?)

What is cryptography?

"cryptography is about communication in the presence of adversaries." - Ron Rivest

“ Tools to help us build secure communication channels that provide:

1) Authentication

2) Integrity

3) Confidentiality

Cryptography As a Tool

- Using cryptography securely is not simple
- Designing cryptographic schemes correctly is near impossible.

Today we want to give you an idea of what can be done with cryptography.

Take a security course if you think you may use it in the future

The Great Divide

	Symmetric Crypto (Private key) (E.g., AES)	Asymmetric Crypto (Public key) (E.g., RSA)
Shared secret between parties?	Yes	No
Speed of crypto operations	Fast	Slow

Symmetric Key: Confidentiality

Motivating Example:

You and a friend share a key K of L random bits, and want to secretly share message M also L bits long.

Scheme:

You send her the *xor(M,K)* and then she "decrypts" using *xor(M,K)* again.

1) Do you get the right message to your friend?
2) Can an adversary recover the message M?
3) Can adversary recover the key K?

Symmetric Key: Confidentiality

- One-time Pad (OTP) is secure but usually impactical
  - Key is as long at the message
  - Keys cannot be reused (why?)

In practice, two types of ciphers are used that require constant length keys:

Stream Ciphers:

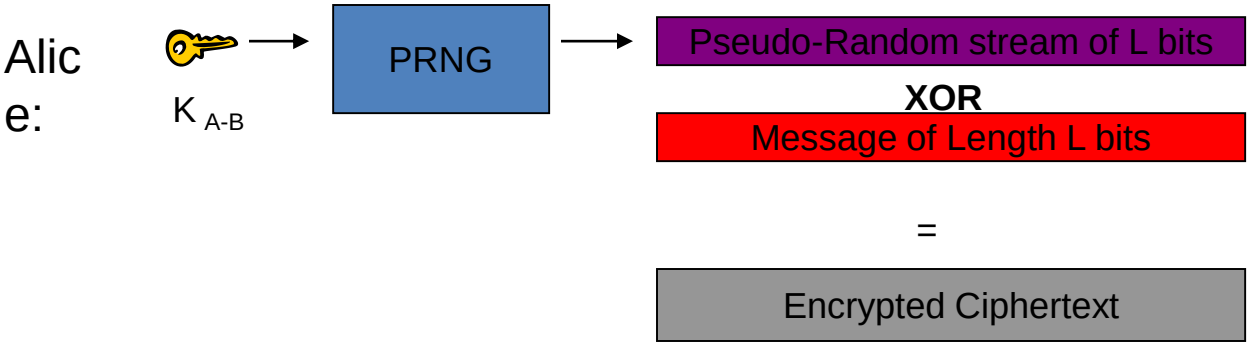
Ex: RC4, A5

Block Ciphers:

Ex: DES, AES, Blowfish

Symmetric Key: Confidentiality

- Stream Ciphers (ex: RC4)



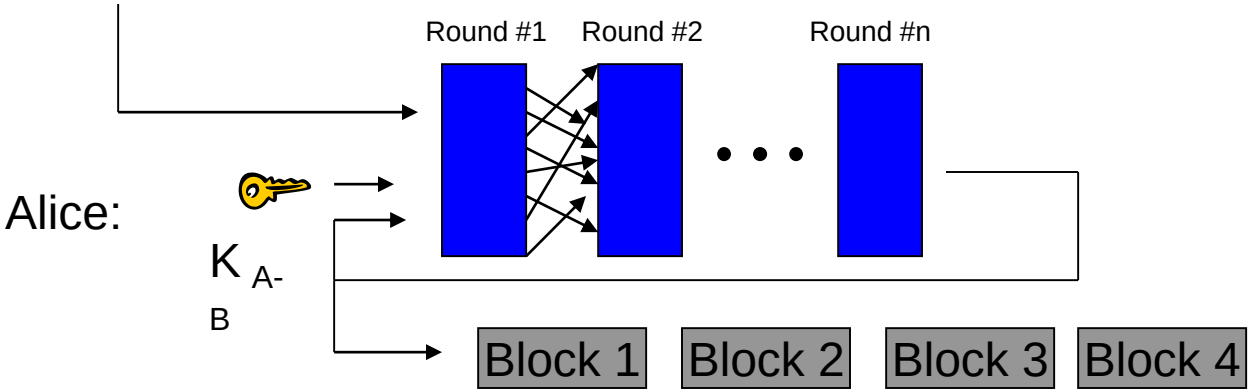
Bob uses K_{A-B} as PRNG seed, and XORs encrypted text to get the message back (just like OTP).

Symmetric Key: Confidentiality

- Block Ciphers (ex: AES)

Block 1 Block 2 Block 3 Block 4

(fixed block size, e.g. 128 bits)



Bob breaks the ciphertext into blocks, feeds it through decryption engine using K_{A-B} to recover the message.

Cryptographic Hash Functions

- Consistent

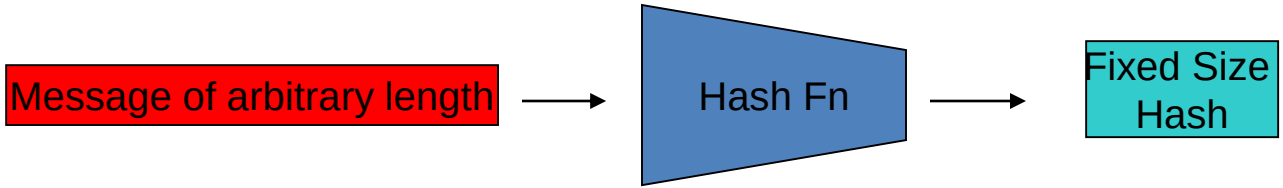
  hash(X) always yields same result

- One-way

  given Y, can't find X s.t. hash(X) = Y

- Collision resistant

  given hash(W) = Z, can't find X such that hash(X) = Z

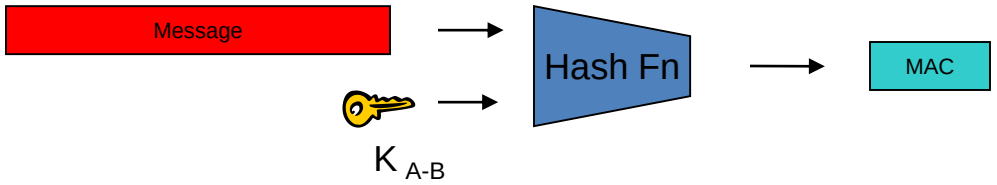


Symmetric Key: Integrity

- Hash Message Authentication Code (HMAC)

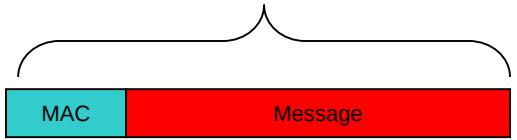
Step #1:

Alice creates MAC



Step #2

Alice Transmits Message & MAC



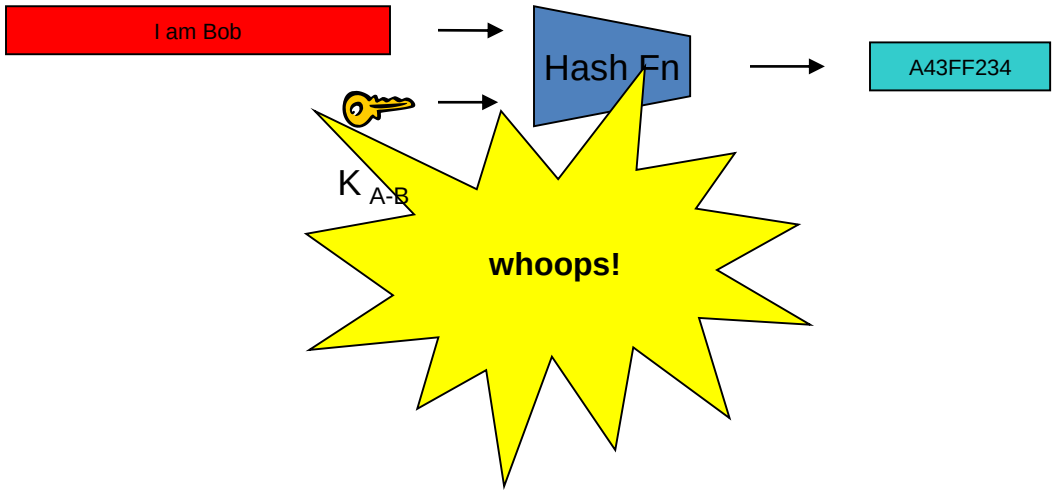
Step #3

Bob computes MAC with message and K_{A-B} to verify.

Why is this secure?
How do properties of a hash function help us?

Symmetric Key: Authentication

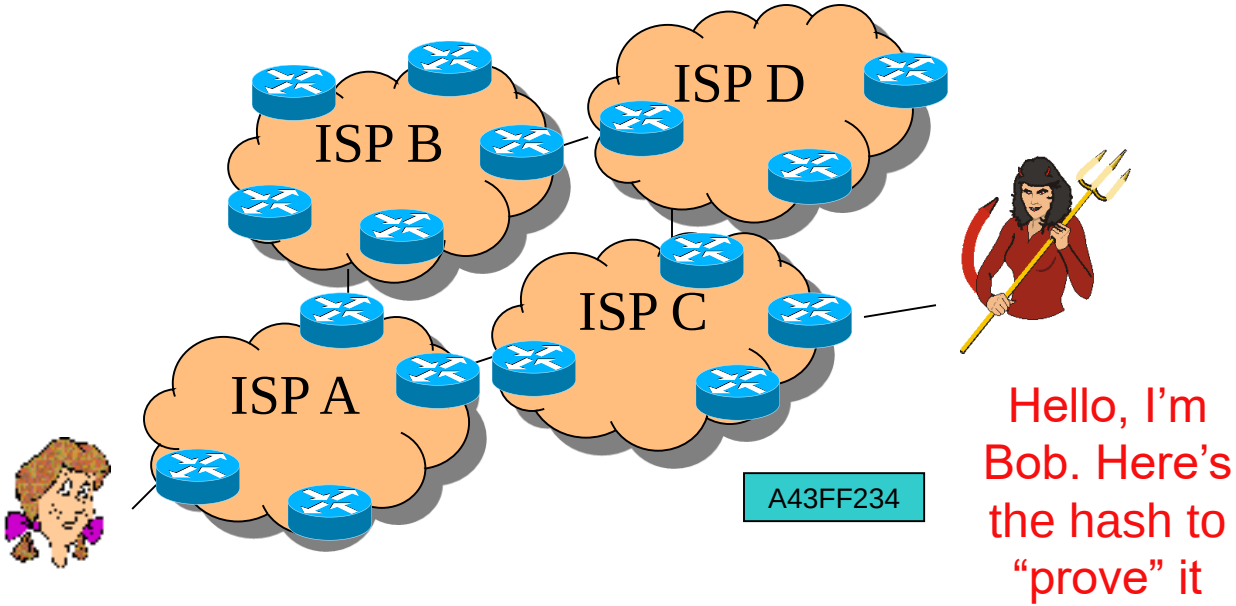
- You already know how to do this!
  (hint: think about how we showed integrity)



Alice receives the hash, computes a hash with K_{A-B} , and she knows the sender is Bob

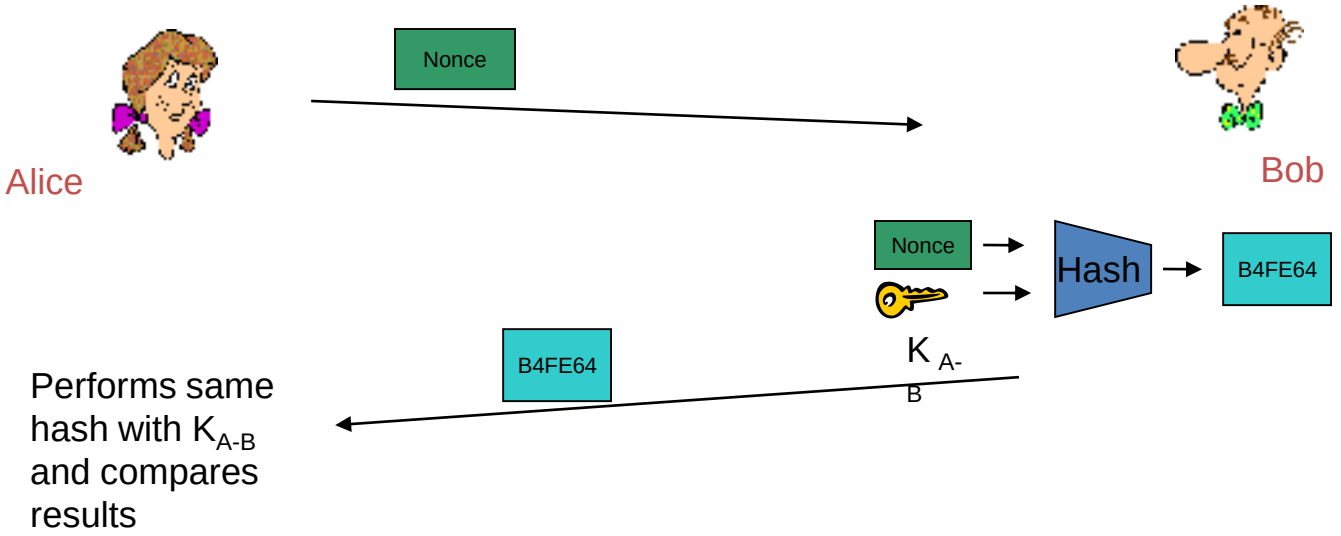
Symmetric Key: Authentication

What if Mallory overhears the hash sent by Bob, and then “replays” it later?



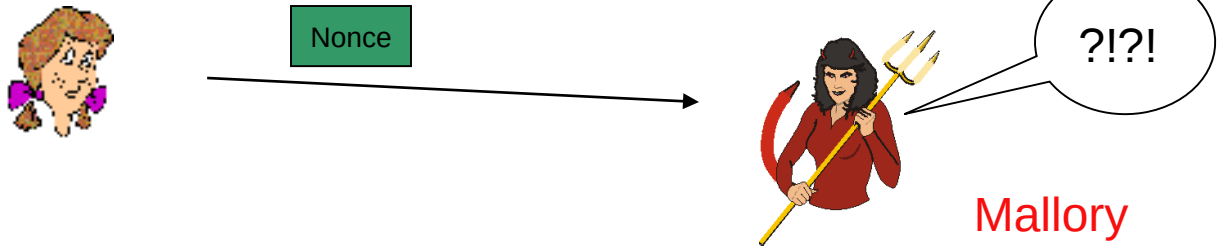
Symmetric Key: Authentication

- A “Nonce”
  - A random bitstring used only once. Alice sends nonce to Bob as a “challenge”. Bob Replies with “fresh” MAC result.



Symmetric Key: Authentication

- A “Nonce”
  - A random bitstring used only once. Alice sends nonce to Bob as a “challenge”. Bob Replies with “fresh” MAC result.



If Alice sends Mallory a nonce, she cannot compute the corresponding MAC without K_{A-B}

Symmetric Key Crypto Review

- Confidentiality: Stream & Block Ciphers
- Integrity: HMAC
- Authentication: HMAC and Nonce

Questions??

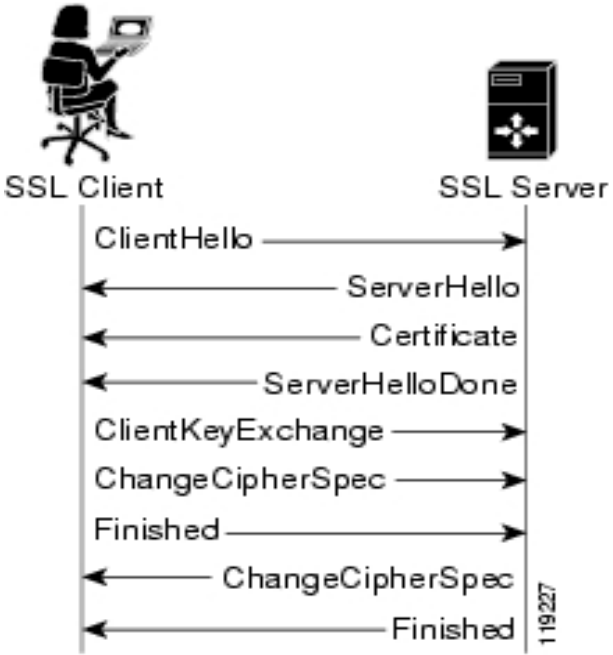
Are we done? Not Really:

1) **Number of keys scales as $O(n^2)$**
2) **How to securely share keys in the first place?**

Transport Layer Security (TLS) aka Secure Socket Layer (SSL)

- Used for protocols like HTTPS
- Special TLS socket layer between application and TCP (small changes to application).
- Handles confidentiality, integrity, and authentication.
- Uses “hybrid” cryptography.

Setup Channel with TLS “Handshake”



Handshake Steps:

1) Client and server negotiate exact cryptographic protocols
2) Client validates public key certificate with CA public key.
3) Client encrypts secret random value with server’s key, and sends it as a challenge.
4) Server decrypts, proving it has the corresponding private key.
5) This value is used to derive symmetric session keys for encryption & MACs.