System and Network Security

Giuseppe Anastasi
g.anastasi@iet.unipi.it
Pervasive Computing & Networking Lab. (PerLab)
Dept. of Information Engineering, University of Pisa

Based on original slides by
- Silberschatz, Galvin and Gagne
- Kurose and Ross

Objectives

PerLab

- Discuss security threats and attacks
- Explain the fundamentals of encryption
- Examine the uses of cryptography in computing
  - Secrecy
  - Message Integrity
  - Digital Signature
  - Authentication
- Describe the various countermeasures to security attacks

Overview

- Threats and attacks
- Cryptography as a Security Tool
  - Secrecy
  - Message integrity
  - Digital signature
  - End-to-end Authentication
  - Secure E-mail
  - Secure Socket Layer (SSL)
- Security Defenses
  - User Authentication
  - Antivirus
  - Firewalls
  - …

Security Threats and Attacks

- Intruders (crackers) attempt to breach security
- **Threat** is potential security violation
- **Attack** is attempt to breach security
- Attack can be accidental or malicious
- Easier to protect against accidental than malicious misuse

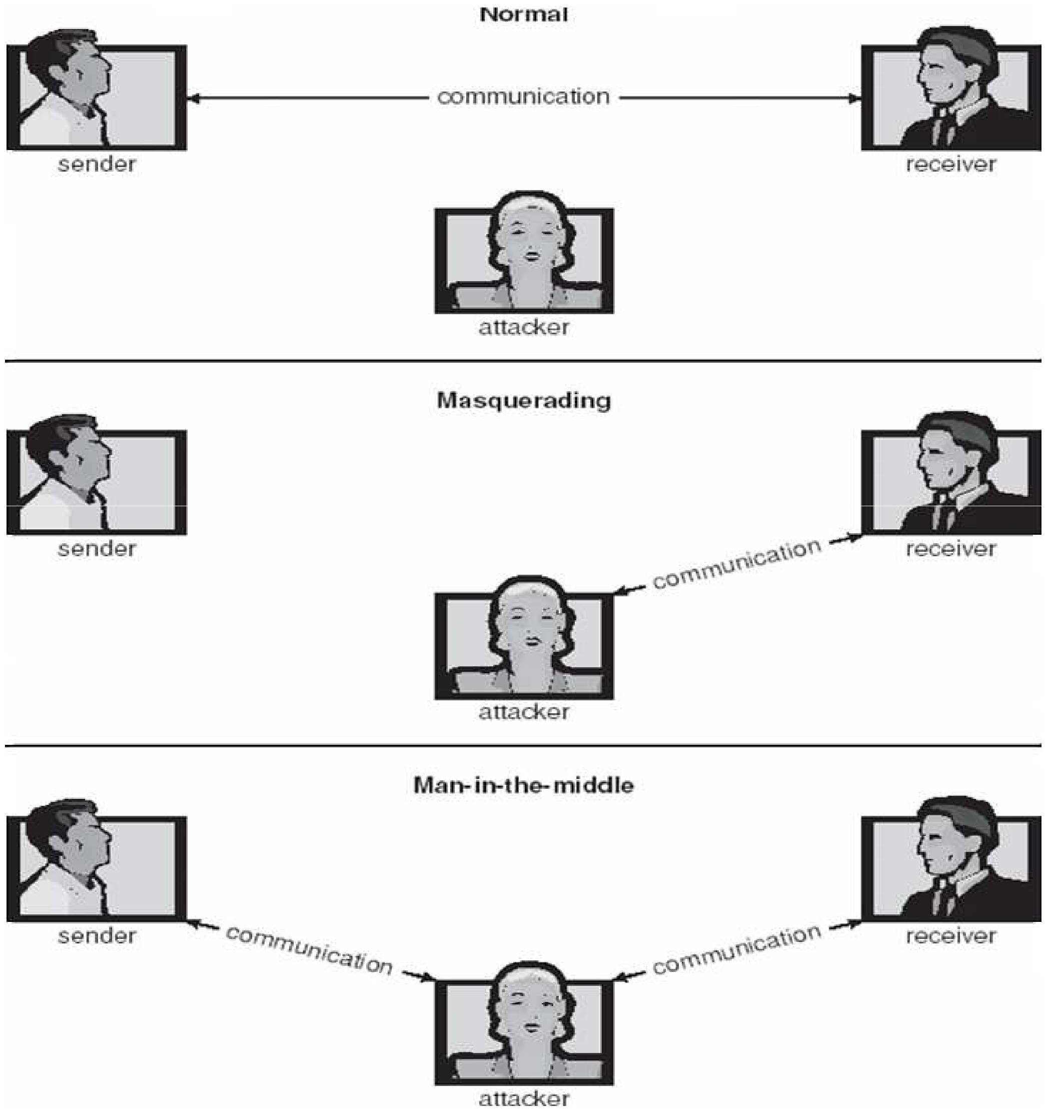
Security Violations

- Categories
  - Breach of confidentiality
  - Breach of integrity
  - Breach of availability
  - Theft of service
  - Denial of service

Security Violations

- Methods
  - Masquerading (breach authentication)
  - Replay attack
    - Message modification
  - Man-in-the-middle attack
  - …

Standard Security Attacks



Security Measure Levels

- Security must occur at four levels to be effective:
  - **Physical**
  - **Human**
    - Avoid social engineering, phishing, dumpster diving
  - **Operating System**
  - **Network**

Security is as weak as the weakest link in the chain

Program Threats

- Trojan Horse
  - Code segment that misuses its environment
  - Exploits mechanisms for allowing programs written by users to be executed by other users
  - Variants:
    - Login spoofing, spyware, pop-up browser windows, covert channels
- Trap Door
  - Specific user identifier or password that circumvents normal security procedures
  - Could be included in a compiler
- Logic Bomb
  - Program that initiates a security incident under certain conditions
- Stack and Buffer Overflow
  - Exploits a bug in a program (overflow either the stack or memory buffers)

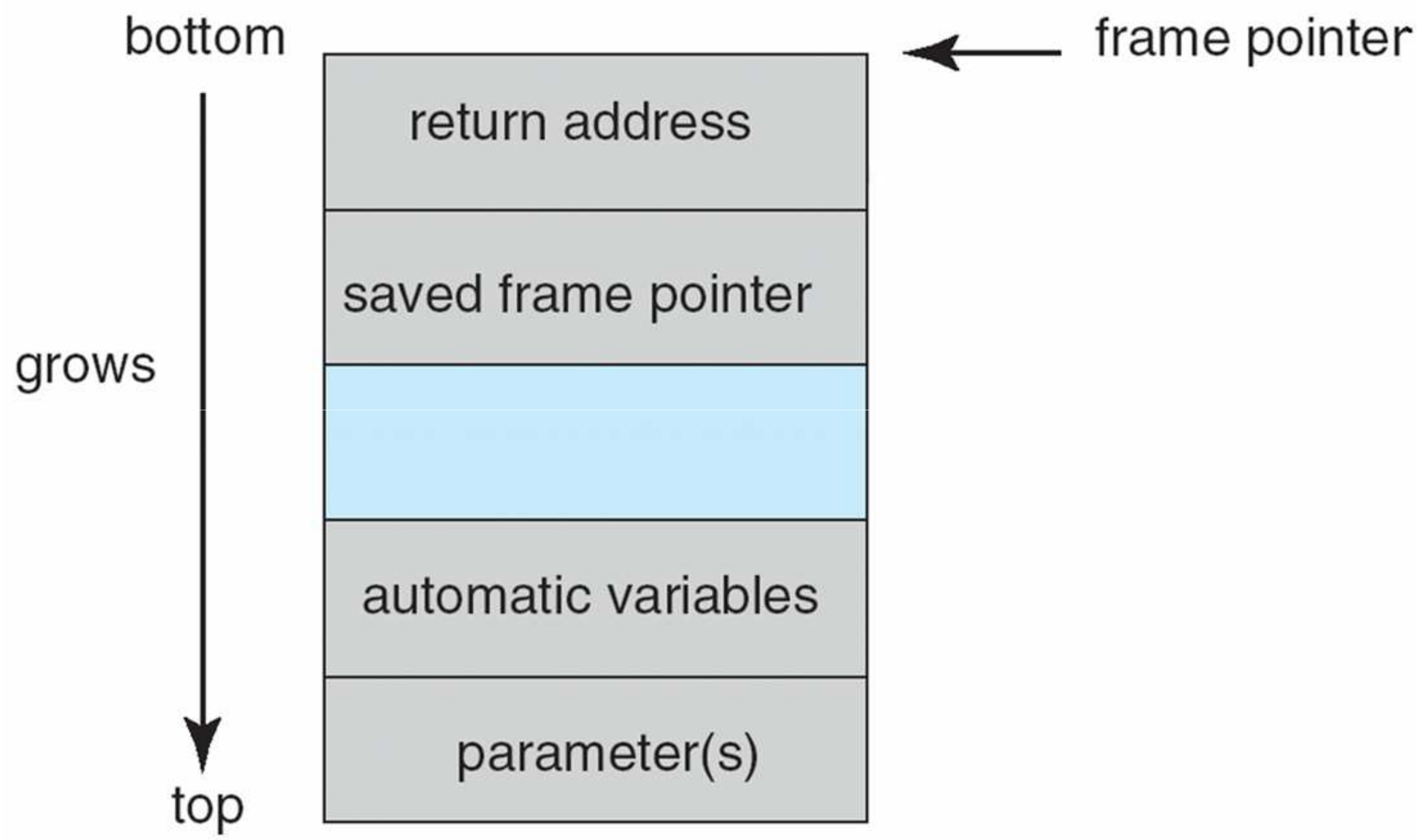
C Program *with* Buffer-overflow Condition

```
#include <stdio.h>
#define BUFFER SIZE 256
int main(int argc, char *argv[])
{
    char buffer[BUFFER SIZE];
    if (argc < 2)
        return -1;
    else {
        strcpy(buffer,argv[1]);
        return 0;
    }
}
```

C Program *without* Buffer-overflow Condition

```
#include <stdio.h>
#define BUFFER SIZE 256
int main(int argc, char *argv[])
{
    char buffer[BUFFER SIZE];
    if (argc < 2)
        return -1;
    else {
        strncpy(buffer, argv[1], sizeof(buffer)-1);
        return 0;
    }
}
```

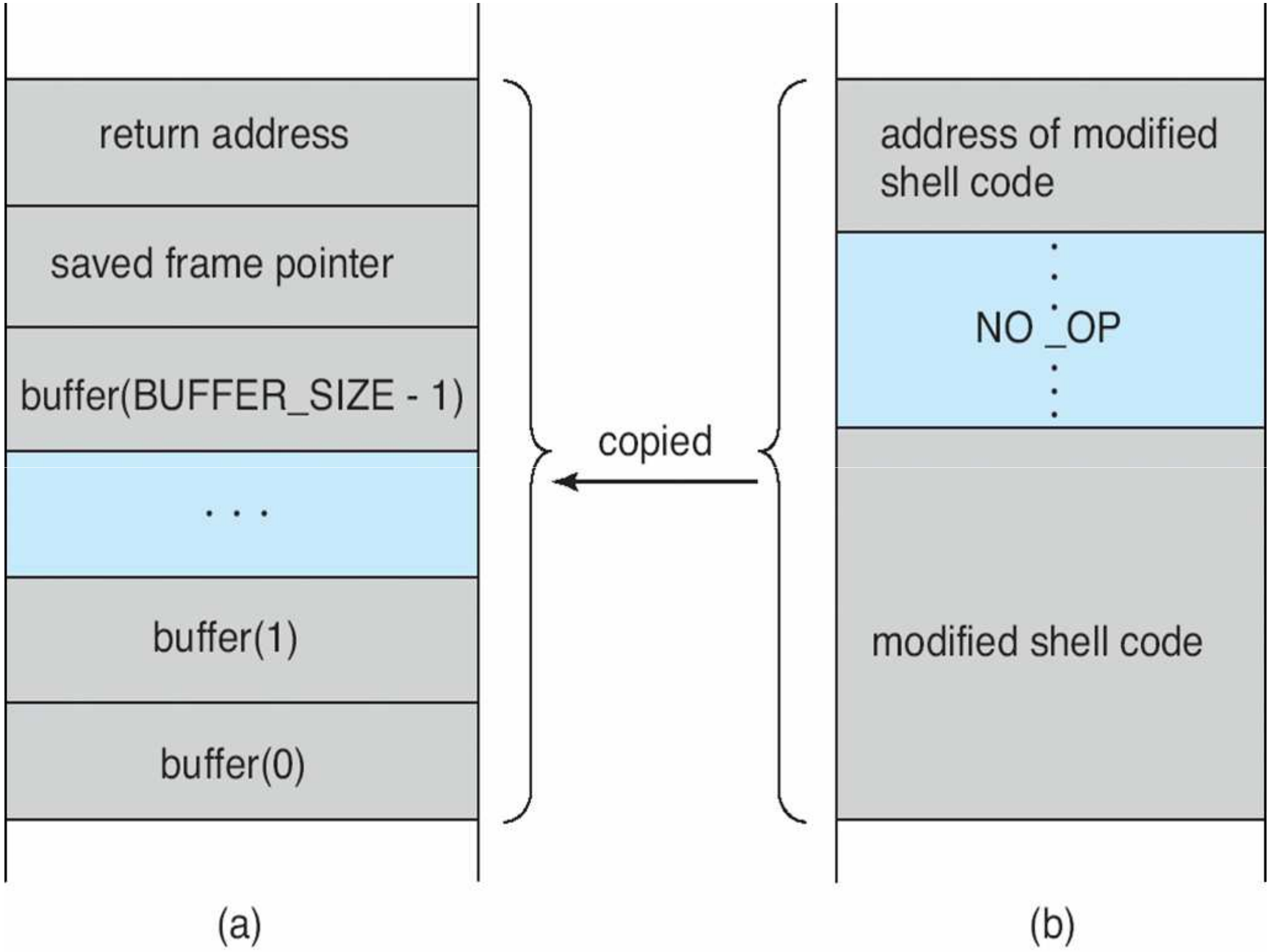
Layout of Typical Stack Frame



Modified Shell Code

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    execvp(“\bin\sh’’,“\bin \sh”, NULL);
    return 0;
}
```

Hypothetical Stack Frame



Before attack

After attack

How to avoid the Buffer-Overflow Attack?

PerLab

- CPU doesn't allow code execution in stack segments
  - Sun Spark, used by Solaris
- NX bit in page table (AMD, Intel)
  - The corresponding page cannot be executed
  - Used by Linux, Windows XP

Program Threats (Cont.)

- Viruses
  - Code fragment embedded in legitimate program
  - Very specific to CPU architecture, operating system, applications
  - Usually borne via email or as a macro
    - Visual Basic Macro to reformat hard drive

```
Sub AutoOpen()
Dim oFS
  Set oFS = CreateObject(’’Scripting.FileSystemObject’’)
  vs = Shell(’’c:command.com /k format     c:’’,vbHide)
End Sub
```

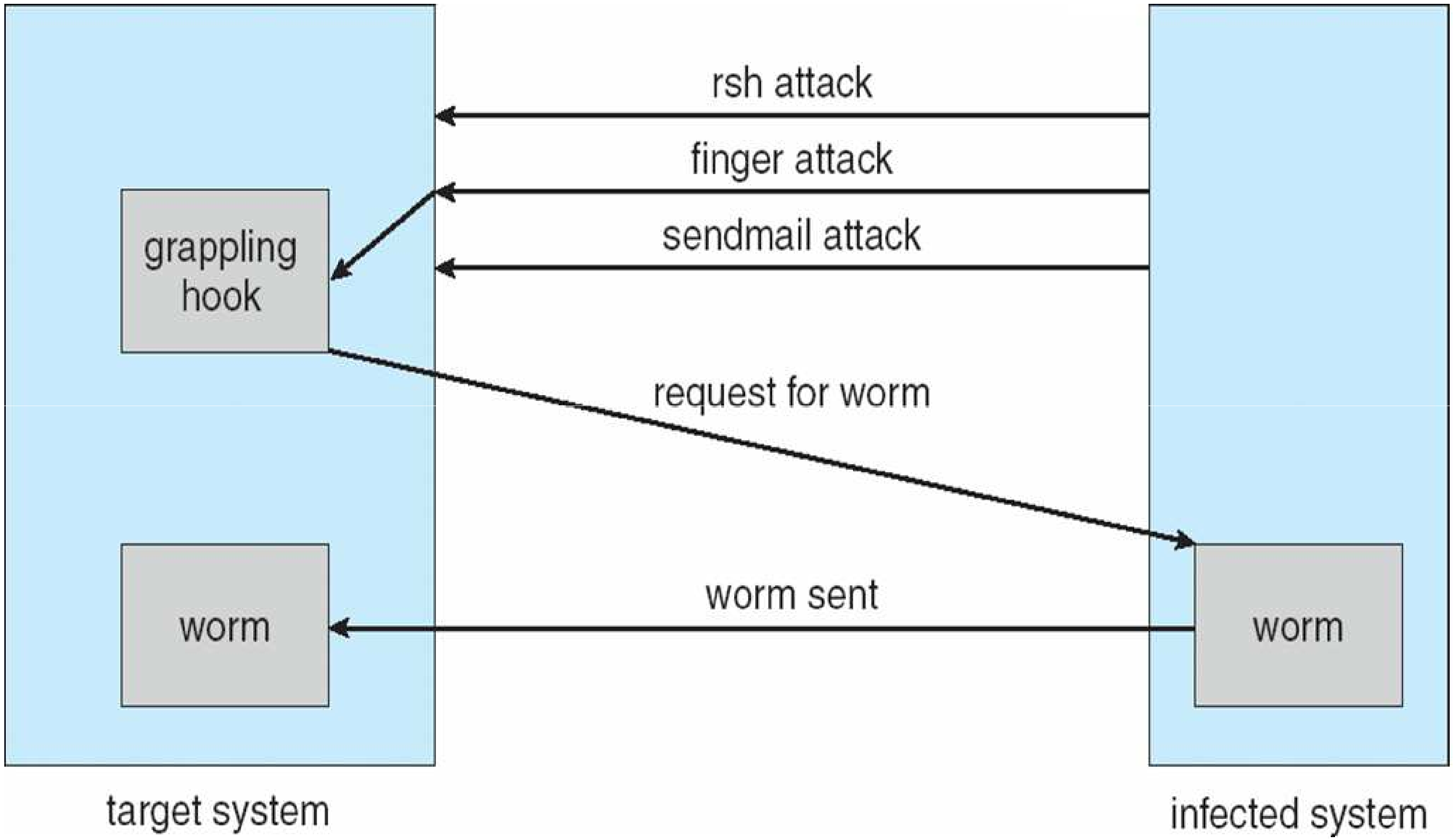
Program Threats (Cont.)

- Virus dropper (typically a Trojan Horse) inserts virus onto the system
- Many categories of viruses, literally thousands of viruses
  - File
  - Boot
  - Macro
  - Source code
  - Polymorphic
  - Encrypted
  - Stealth (clandestino)
  - Tunneling (sotterraneo)
  - Multipartite (composito
  - Armored (corazzato)

System and Network Threats

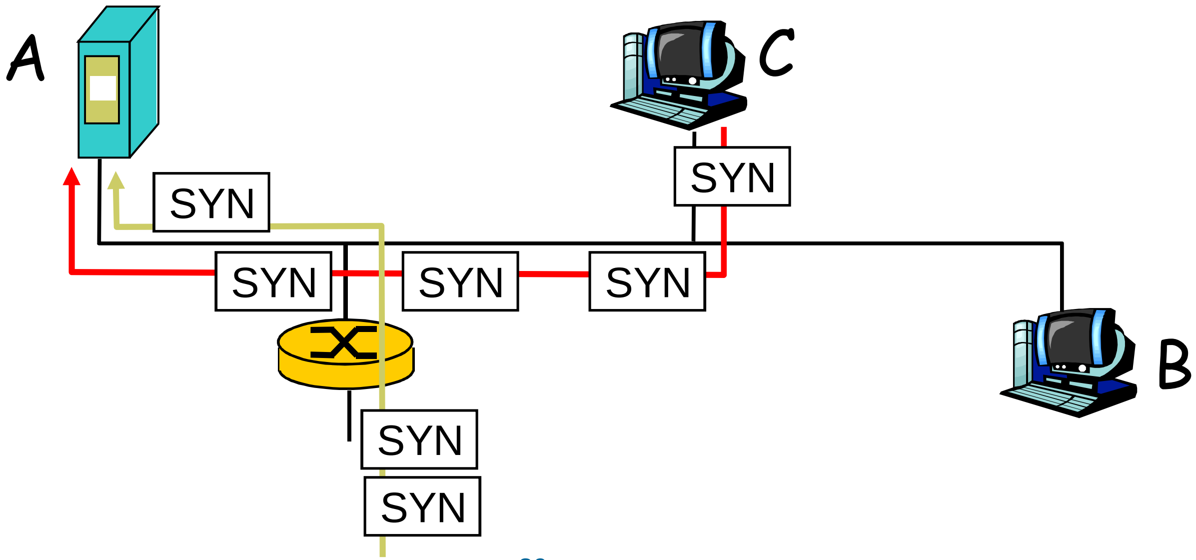
- Worms
  - use spawn mechanism; standalone program
- Morris Internet worm (2 Nov 1988)
  - Exploited UNIX networking features (remote access) and bugs in *finger* and *sendmail* programs
  - Grappling hook program uploaded main worm program
- Port scanning
  - Automated attempt to connect to a range of ports on one or a range of IP addresses

The Morris Internet Worm



System and Network Threats

- Denial of Service
  - Overload the targeted computer preventing it from doing any useful work
  - Distributed denial-of-service (DDOS) come from multiple sites at once
  - SYN Flooding



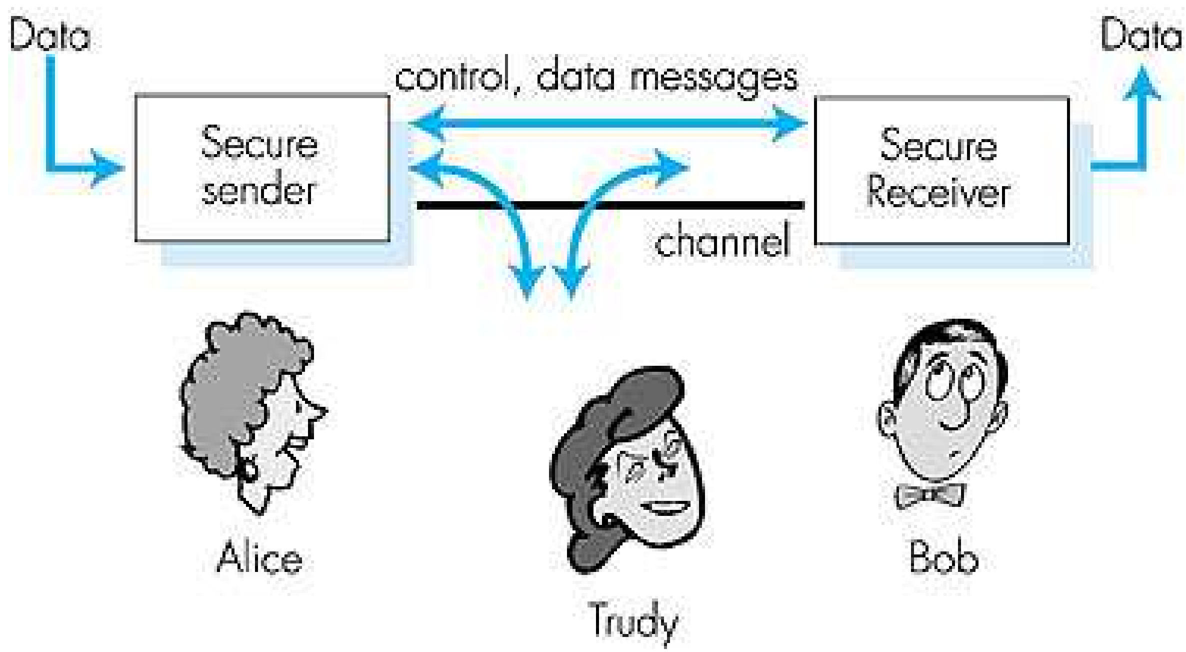
Overview

- Threats and attacks
- Cryptography as a Security Tool
  - Secrecy
  - Message integrity
  - Digital signature
  - End-to-end Authentication
  - Secure E-mail
  - Secure Socket Layer (SSL)
- Security Defenses
  - User Authentication
  - Antivirus
  - Firewalls
  - …

Cryptography as a Security Tool

- Broadest security tool available
  - Source and destination of messages cannot be trusted without cryptography
  - Means to constrain potential senders (*sources*) and / or receivers (*destinations*) of *messages*
- Allows *secure communications* over an intrinsically *insecure medium*

Friends and Enemies: Alice, Bob, Trudy



- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy, the "intruder" may intercept, delete, add messages

What does secure communication mean?

PerLab

Secrecy: only sender, intended receiver should "understand" msg contents

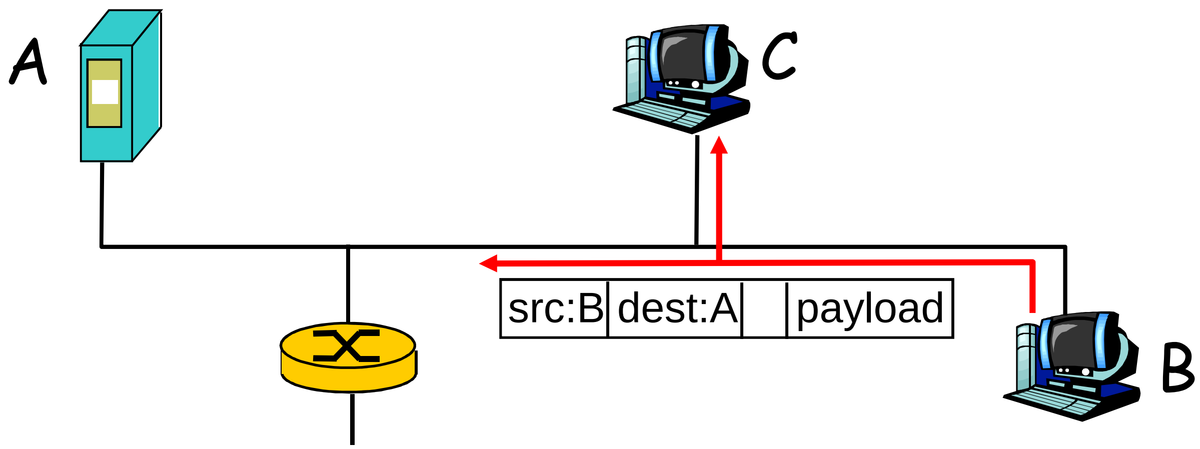
- sender encrypts msg
- receiver decrypts msg

Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

End-to-end Authentication: sender, receiver want to confirm identity of each other

Insecure communication medium

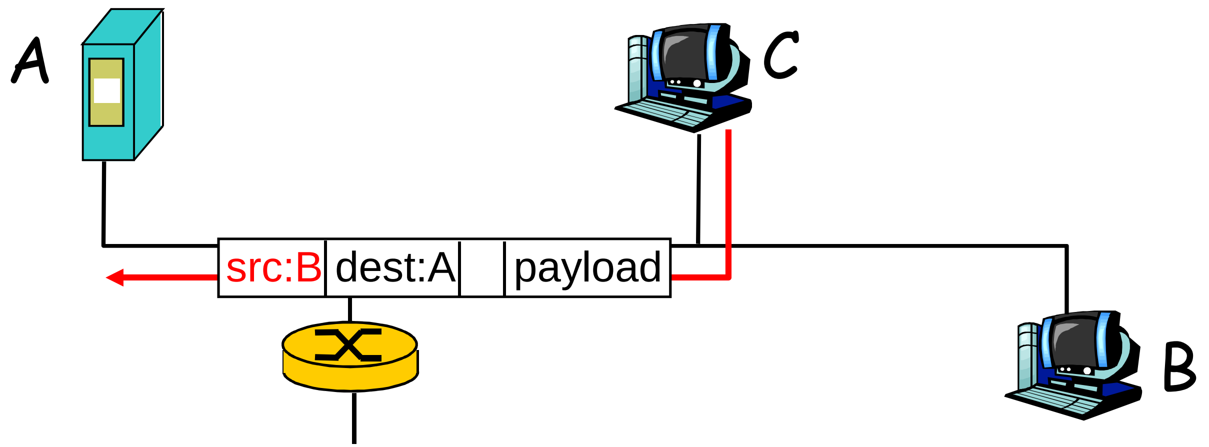
- Packet sniffing:
  - broadcast media
  - promiscuous NIC reads all packets passing by
  - can read all unencrypted data (e.g. passwords)
  - e.g.: C sniffs B's packets



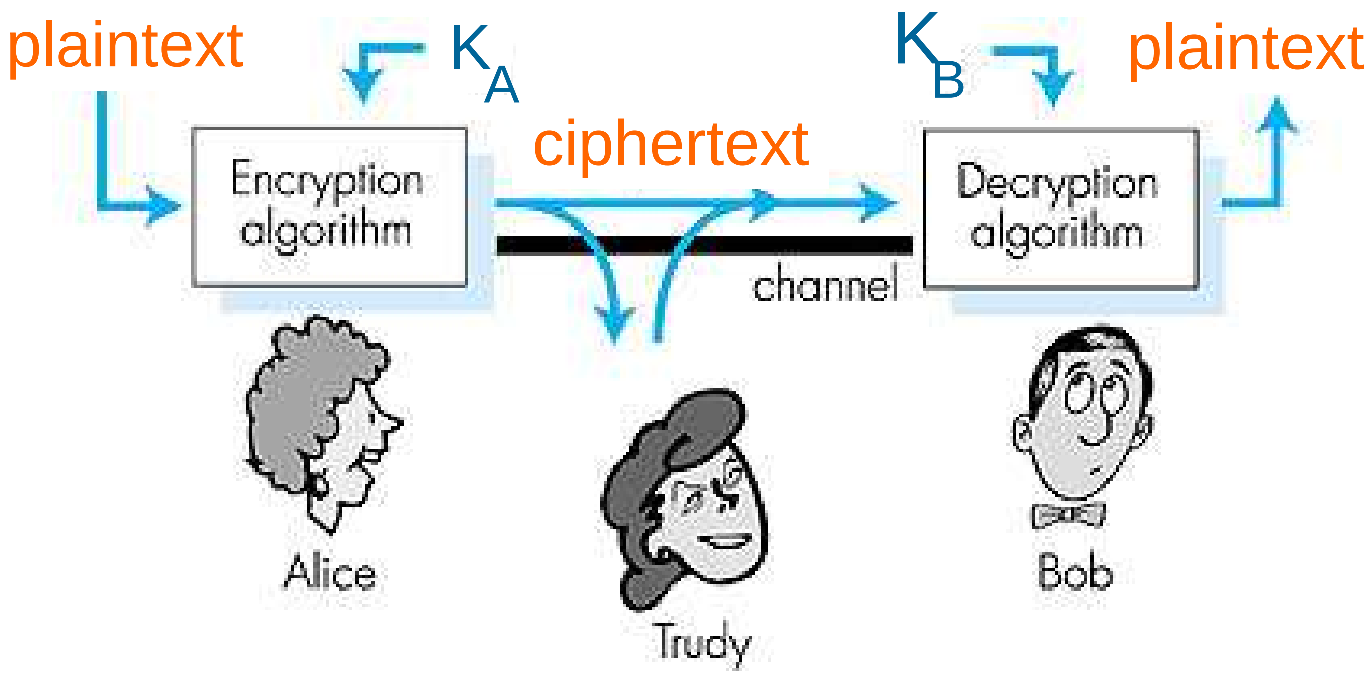
Insecure communication medium

IP Spoofing

- can generate "raw" IP packets directly from application, putting any value into IP source address field
- receiver can't tell if source is spoofed
- e.g.: C pretends to be B



The language of cryptography



symmetric key crypto: sender, receiver keys identical

public-key crypto: encrypt key *public*, decrypt key *secret*

Who might Bob, Alice be?

- … well, *real-life* Bobs and Alices (e.g., lovers)!
- Web browser/server for electronic transactions
  - e.g., on-line purchases
- on-line banking client/server
- E-mail programs
- DNS servers
- routers exchanging routing table updates
- other examples?

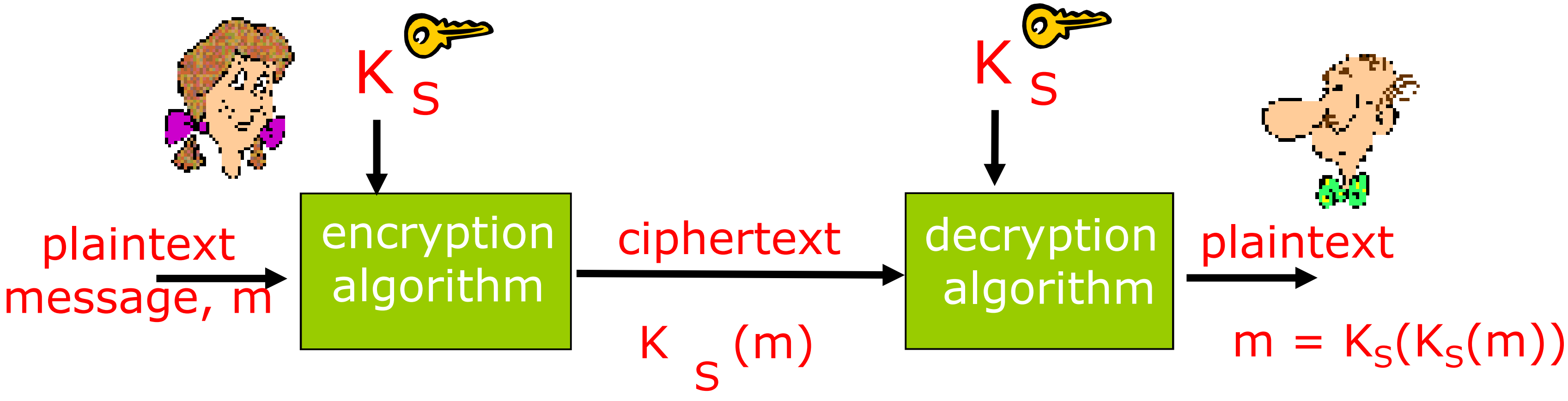
Overview

- Threats and attacks
- Cryptography as a Security Tool
  - Secrecy
  - Message integrity
  - Digital signature
  - End-to-end Authentication
  - Secure E-mail
  - Secure Socket Layer (SSL)
- Security Defenses
  - User Authentication
  - Antivirus
  - Firewalls
  - …

Types of Cryptography

- Crypto often uses keys:
  - Algorithm is known to everyone
  - Only “keys” are secret
- Public key cryptography
  - Involves the use of two keys
- Symmetric key cryptography
  - Involves the use of one key
- Hash functions
  - Involves the use of no keys
  - Nothing secret: How can this be useful?

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

Cesar cypher

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:  abcdefghijklmnopqrstuvwxyz

ciphertext: ghijklmnopqrstuvwxyzabcdef
```

E.g.:

```
Plaintext:  bob. i love you. alice
ciphertext: huh. o rubk eua. groik
```

Key: offset between the character in the pain text and the corresponding character in the cyphertext

Monoalphabetic cypher

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz

ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

E.g.:
```
 Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

Key: the mapping from the set of 26 letters to the set of 26 letters

Poly-alphabetic encryption

- n monoalphabetic cyphers, $M_1, M_2, \ldots, M_n$
- Cycling pattern:
  - e.g., n=4, M_1,M_3,M_4,M_3,M_2; M_1,M_3,M_4,M_3,M_2;
- For each new plaintext symbol, use subsequent monoalphabetic pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4
- Key: the n ciphers and the cyclic pattern

Breaking an encryption scheme

- Cipher-text only attack:
  - Trudy has ciphertext that she can analyze
- Two approaches:
  - Search through all keys
  - Statistical analysis

- Known-plaintext attack:
  - trudy has some plaintext corresponding to some ciphertext
  - eg, in monoalphabetic cipher, trudy determines pairings for a,l,i,c,e,b,o,
- Chosen-plaintext attack
  - trudy can get the cyphertext for some chosen plaintext

Two types of symmetric ciphers

- Stream ciphers
  - encrypt one bit at time
- Block ciphers
  - Break plaintext message in equal-size blocks
  - Encrypt each block as a unit

DES: Data Encryption Standard

PerLab

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- Block cipher with cipher block chaining
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - No known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

  (actually encrypt, decrypt, encrypt)

AES: Advanced Encryption Standard

- new (Nov. 2001) symmetric-key NIST standard, replacing DES
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Key Question

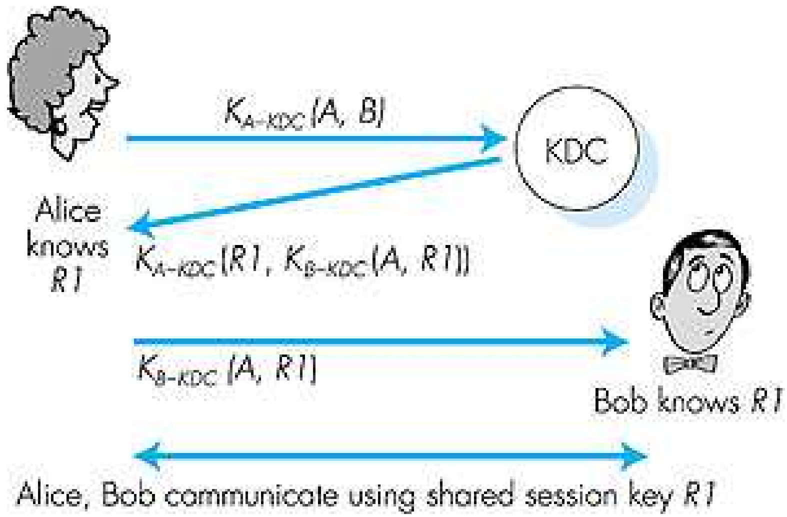
How do two entities establish shared secret key over network?

Solutions:

- Direct exchange (in person)
- Key Distribution Center (KDC)
  - Trusted entity acting as intermediary between entities
- Using public key cryptography

Key Distribution Center (KDC)

- Alice,Bob need shared symmetric key.
- KDC: server shares different secret key with each registered user.
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.



- Alice communicates with KDC, gets session key R1, and $K_{B\text{-}KDC}$(A,R1)
- Alice sends Bob $K_{B\text{-}KDC}$(A,R1), Bob extracts R1
- Alice, Bob now share the symmetric key R1.

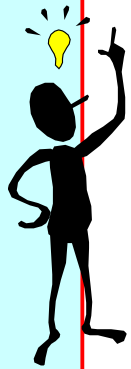
Public Key Cryptography

symmetric key crypto

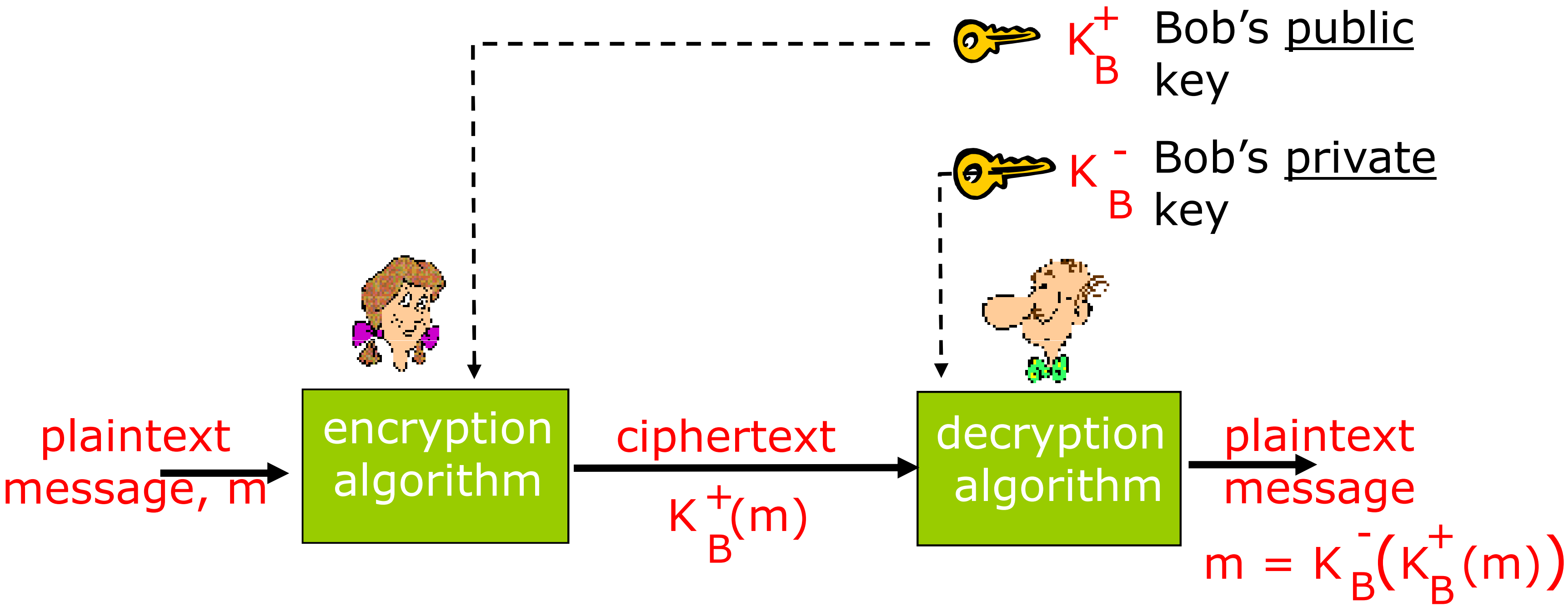
- requires sender, receiver know shared secret key
- How to agree on key in first place
  - particularly if never “met”?

public key cryptography

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver



Public key cryptography



Public key encryption algorithms

Requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$m = K_B^-(K_B^+(m))$$

2. given the public key, it should be impossible to compute private key

RSA: Rivest, Shamir, Adelson algorithm

RSA: another important property

The following property will be *very* useful later:

$$m = K_B^-(K_B^+(m))$$ $$m = K_B^+(K_B^-(m))$$

$\underbrace{\qquad\qquad}$ use public key first, followed by private key

$\underbrace{\qquad\qquad}$ use private key first, followed by public key

Result is the same!

44

Session keys

- Public key cryptography is computationally intensive
- DES is at least 100 times faster than RSA

Session key, K_S

- Bob and Alice use RSA to exchange a symmetric key K_S
- Once both have K_S, they use symmetric key cryptography

Overview

- Threats and attacks
- Cryptography as a Security Tool
    - Secrecy
    - Message integrity
    - Digital signature
    - End-to-end Authentication
    - Secure E-mail
    - Secure Socket Layer (SSL)
- Security Defenses
    - User Authentication
    - Antivirus
    - Firewalls
    - …

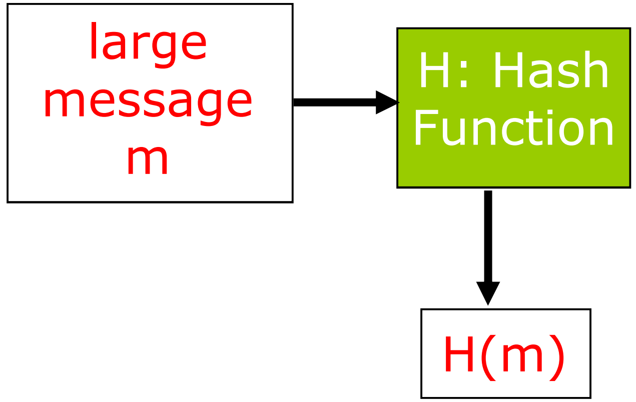
Message Integrity

PerLab

- Allows communicating parties to verify that received messages are authentic.
  - Source of message is who/what you think it is
  - Content of message has not been altered
  - Message has not been replayed
  - Sequence of messages is maintained
- Let's first talk about message digests

Message Digests

- Function H( ) that takes as input an arbitrary length message and outputs a fixed-length string: “message signature”
- Note that H( ) is a many-to-1 function
- H( ) is often called a “hash function”



- Desirable properties:
  - Easy to calculate
  - Irreversibility: Can’t determine m from H(m)
  - Collision resistance: Given [m, H(m)], it must be computationally unfeasible to produce m’ (with m<>m’) such that H(m) = H(m’)
  - Seemingly random output

Internet checksum

Internet checksum has some properties of hash function:

- produces fixed length digest (16-bit sum) of input
- is many-to-one

- But given message with given hash value, it is easy to find another message with same hash value.
- Example: Simplified checksum: add 4-byte chunks at a time:

message	ASCII format
I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 D2 42
	B2 C1 D2 AC

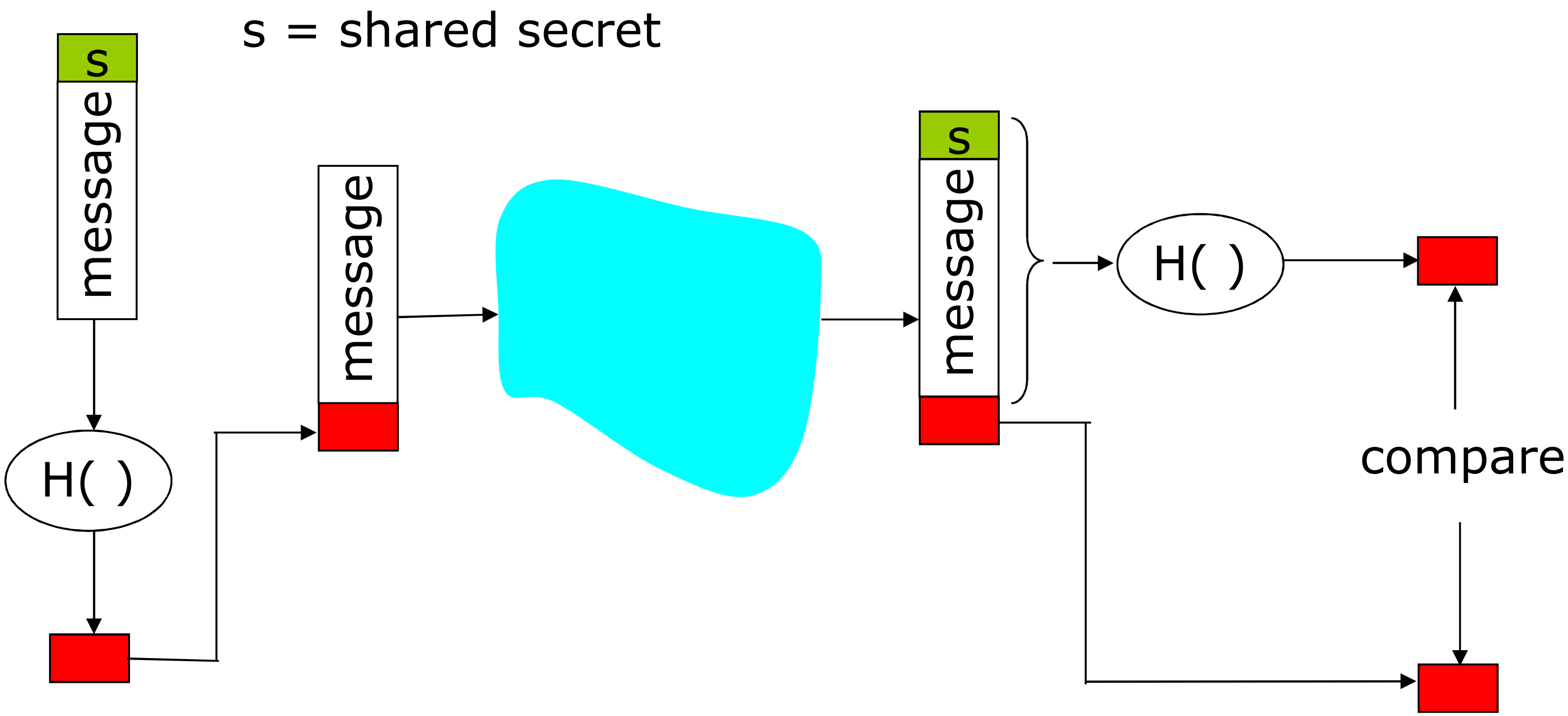
message	ASCII format
I O U 9	49 4F 55 31
0 0 . 1	30 30 2E 39
9 B O B	39 42 D2 42
	B2 C1 D2 AC

different messages but identical checksums!

Hash Function Algorithms

- MD5 hash function widely used [Rivest, RFC 1321]
  - computes 128-bit message digest in 4-step process.
  - C source code implementation available in RFC 1321
- SHA-1 is also used.
  - US standard [NIST]
  - 160-bit message digest

Message Authentication Code (MAC)



- ***Authenticates sender***
- ***Verifies message integrity***
- Sender:
  - calculates MAC: H(m||s) ;
  - send [m|| H(m||s)]
- No encryption ! Also called “keyed hash”

HMAC [RFC 2104]

- Popular MAC standard
- Can use both MD5 and SHA-1

1. Concatenates secret to front of message: [s||m]
2. Hashes concatenated message: H([s||m])
3. Concatenates the secret to front of digest: [H([s||m])||m]
4. Hashes the combination again: H([H([s||m])||m])

Overview

- Threats and attacks
- Cryptography as a Security Tool
  - Secrecy
  - Message integrity
  - Digital signature
  - End-to-end Authentication
  - Secure E-mail
  - Secure Socket Layer (SSL)
- Security Defenses
  - User Authentication
  - Antivirus
  - Firewalls
  - …

Digital Signature

- Cryptographic technique analogous to hand-written signatures.
  - The sender (Bob) digitally signs document, establishing he is the document owner/creator.
- Verifiable
  - The recipient (Alice) can verify and prove that Bob, and no one else, signed the document.
- Non-forgeable
  - The sender (Bob) can prove that someone else has signed a message
- Non repudiation
  - The recipient (Alice) can prove that Bob signed m and not m’
- Message integrity
  - The sender (Bob) can prove that he signed m and not m’

Digital Signatures

Could we use Message Authentication Code as a Digital Signature??

- Goal is similar to that of a MAC
  - MAC guarantees message integrity
- MAC does not guarantee
  - Verifiability
  - Non forgeability
  - Non repudiation
- Solution: use public key cryptography

Digital Signatures

Simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B^-, creating “signed” message, $K_B^-(m)$

Bob’s message, m

Dear Alice

Oh, how I have missed you. I think of you all the time! …(blah blah blah)

Bob

K_B^- Bob’s private key

Public key encryption algorithm

$K_B^-(m)$

Bob’s message, m, signed (encrypted) with his private key

Digital Signatures (more)

- Suppose Alice receives msg m, digital signature $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$, then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

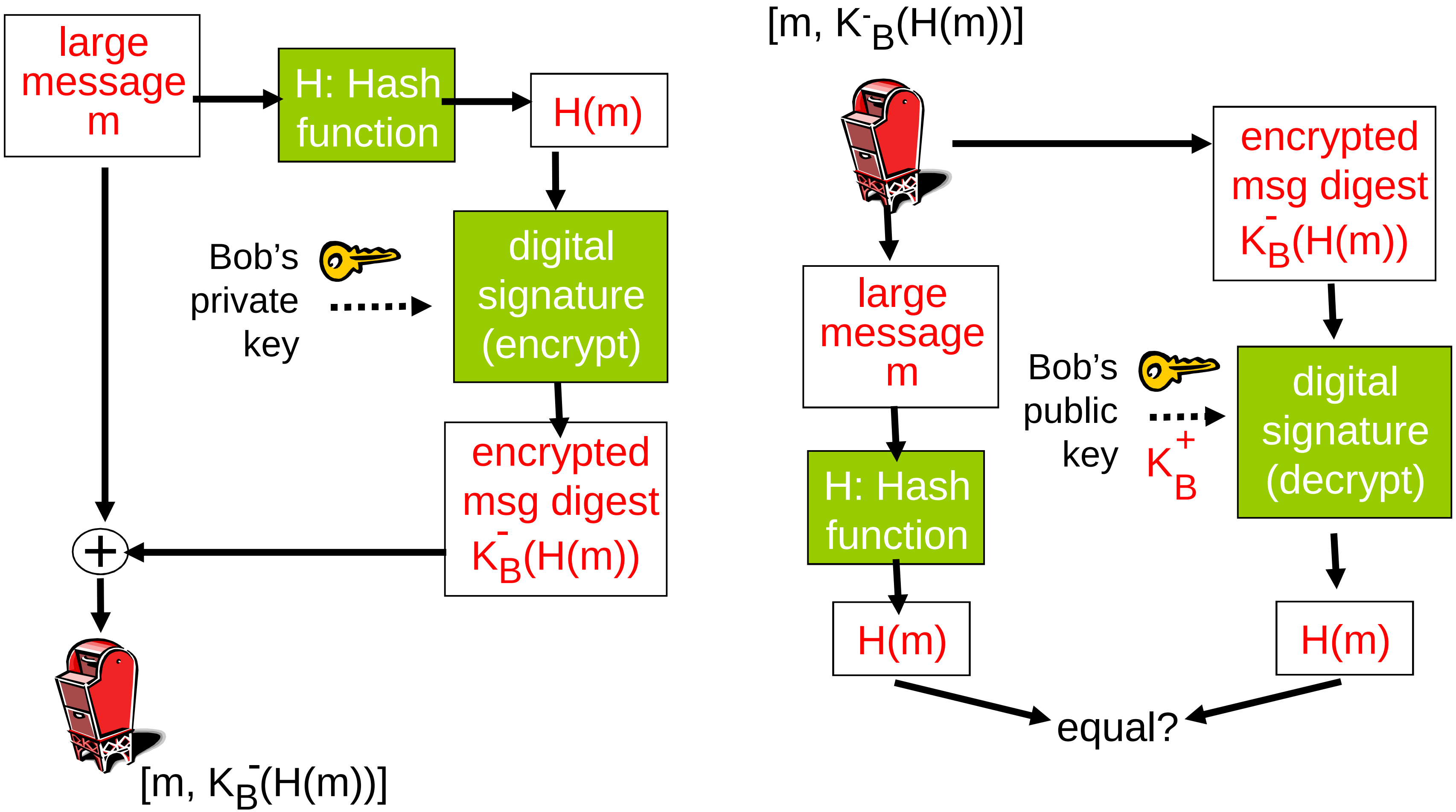
Are requirements satisfied?

- Alice thus verifies that:
  - Bob signed *m*.
  - No one else signed *m*.
  - Bob signed m and not *m'*.
- Non-repudiation:
  - Alice can take *m*, and signature $K_B^-(m)$ to court and prove that Bob signed *m*.
- Message Integrity
  - Bob can prove that he signed m and not *m'*.

Signed message digest

Bob sends digitally signed message:

Alice verifies signature and integrity of digitally signed message:



Authentication Code vs. Digital Signature

- MAC: m+s → H(m+s) → [m, H(m+s)]
- DS: m → H(m) → K^-(H(m)) → [m, K^-(H(m))]
- Digital signature is a heavier technique
  - Requires a Public Key Infrastructure (PKI)
- In practice
  - MAC used in OSPF for message integrity
  - MAC also used for transport and network layer solutions
  - DS used in PGP for message integrity and non repudiation

Key Question

PerLab

- How can Alice achieve Bob's public key?
  - E-mail?
  - Website?
  - ??

61

Motivation for public-key certification

- Trudy send a message to Bob
  - Trudy creates e-mail message:

    My loved Bob,

    I also think of you all the time!

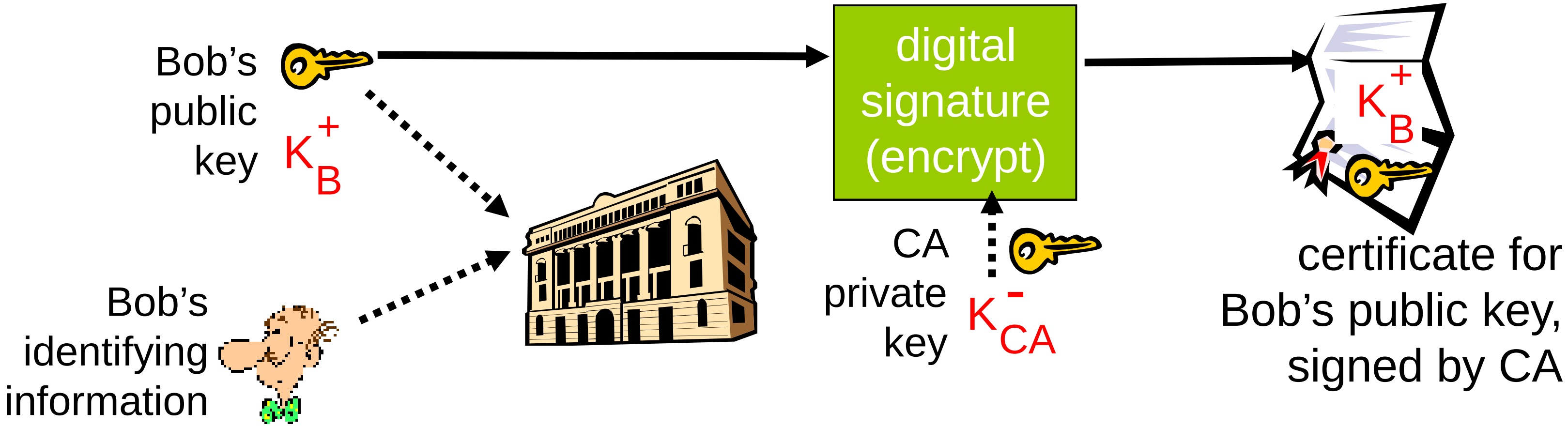
    I want to take you in marriage asap!

    Alice

  - Trudy signs message with her private key
  - Trudy sends message to Bob
  - Trudy sends Bob her public key, but says it's Alice's public key.
  - Bob verifies signature
  - Bob assumes that message is authentic

Certification Authorities

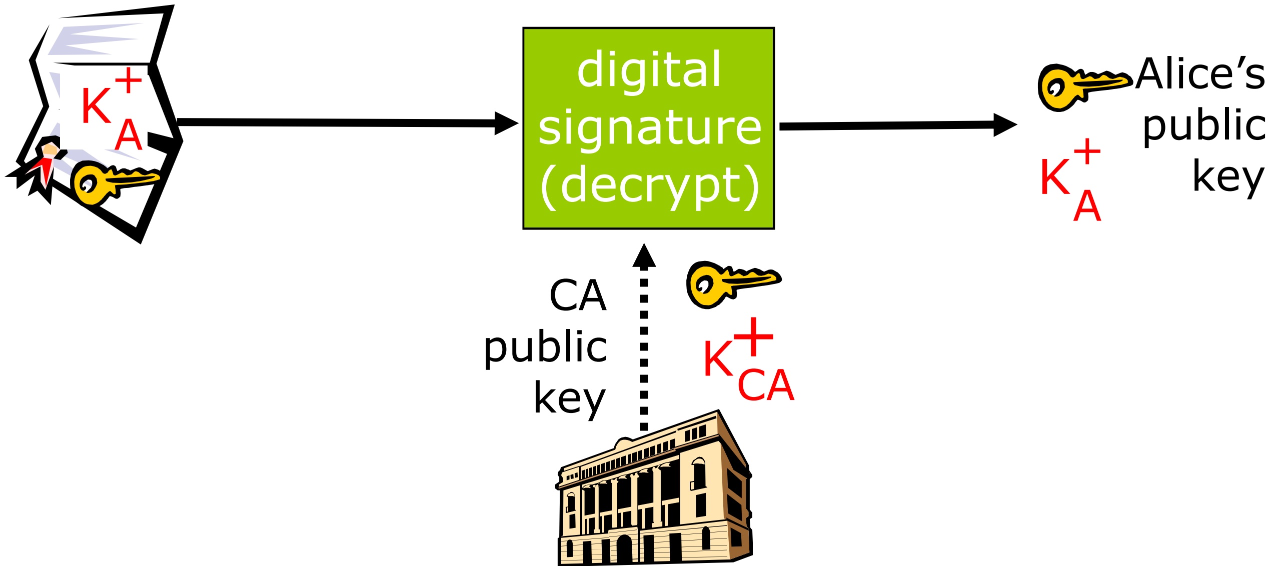
- Certification authority (CA):
  - binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides “proof of identity” to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E’s public key digitally signed by CA – CA says “this is E’s public key”



63

Certification Authorities

- When Bob wants Alice's public key:
  - gets Alice's certificate (even from Alice).
  - apply CA's public key to Alice's certificate, get Alice's public key



Certificates

- Primary standard ITU X.509 (RFC 2459)
- Certificate includes:
  - Issuer name
  - Entity name, address, domain name, etc.
  - Entity’s public key
  - Digital signature (signed with issuer’s private key)
- Public-Key Infrastructure (PKI)
  - Certificates and certification authorities
  - Often considered “heavy”

Overview

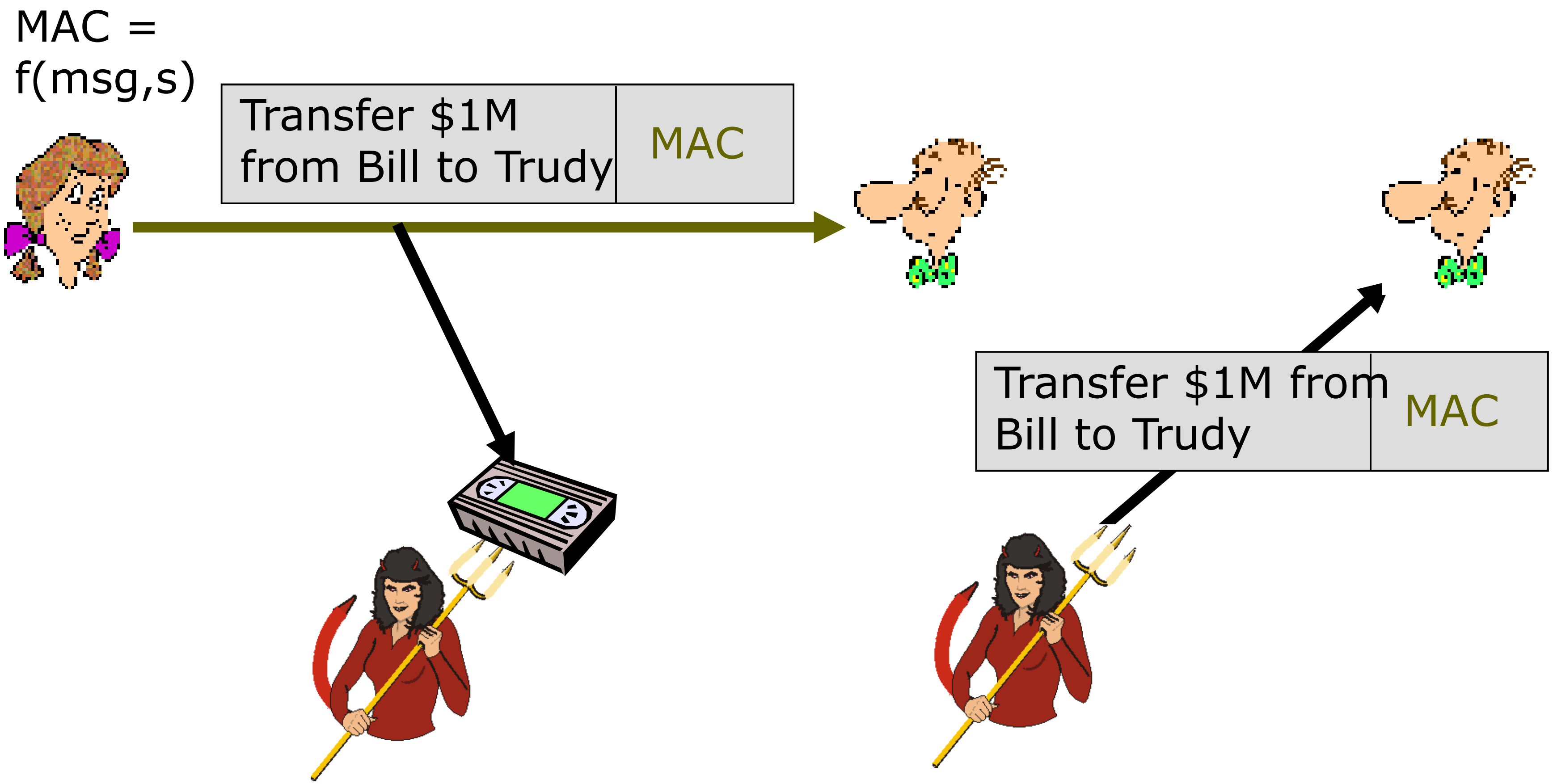
PerLab

- Threats and attacks
- Cryptography as a Security Tool
  - Secrecy
  - Message integrity
  - Digital signature
  - End-to-end Authentication
  - Secure E-mail
  - Secure Socket Layer (SSL)
- Security Defenses
  - User Authentication
  - Antivirus
  - Firewalls
  - …

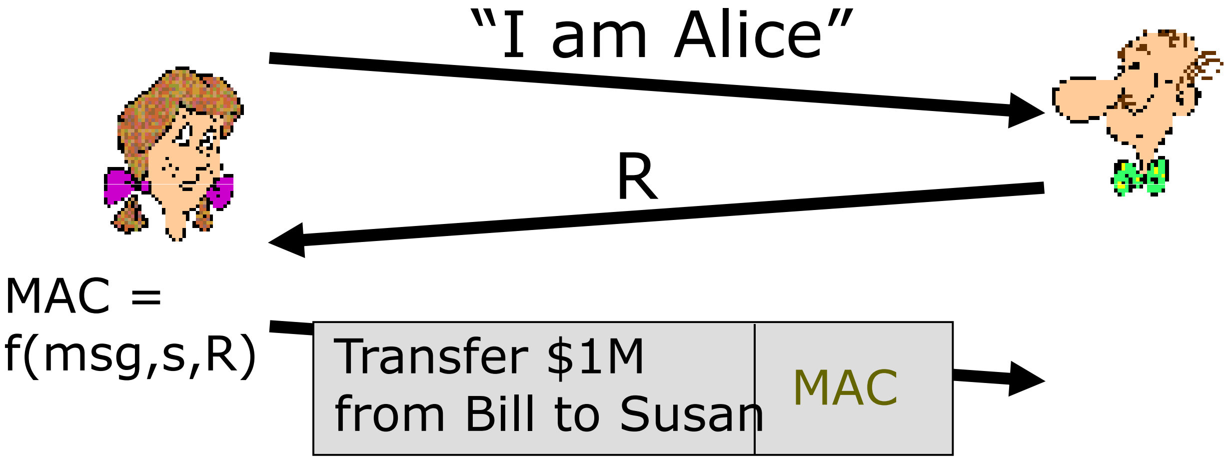
End-point authentication

- Want to be sure of the originator of the message – *end-point authentication*.
- Assuming Alice and Bob have a shared secret, will MAC provide end-point authentication?
  - We do know that Alice created the message.
  - But did she send it?

Playback attack



Defending against playback attack: nonce

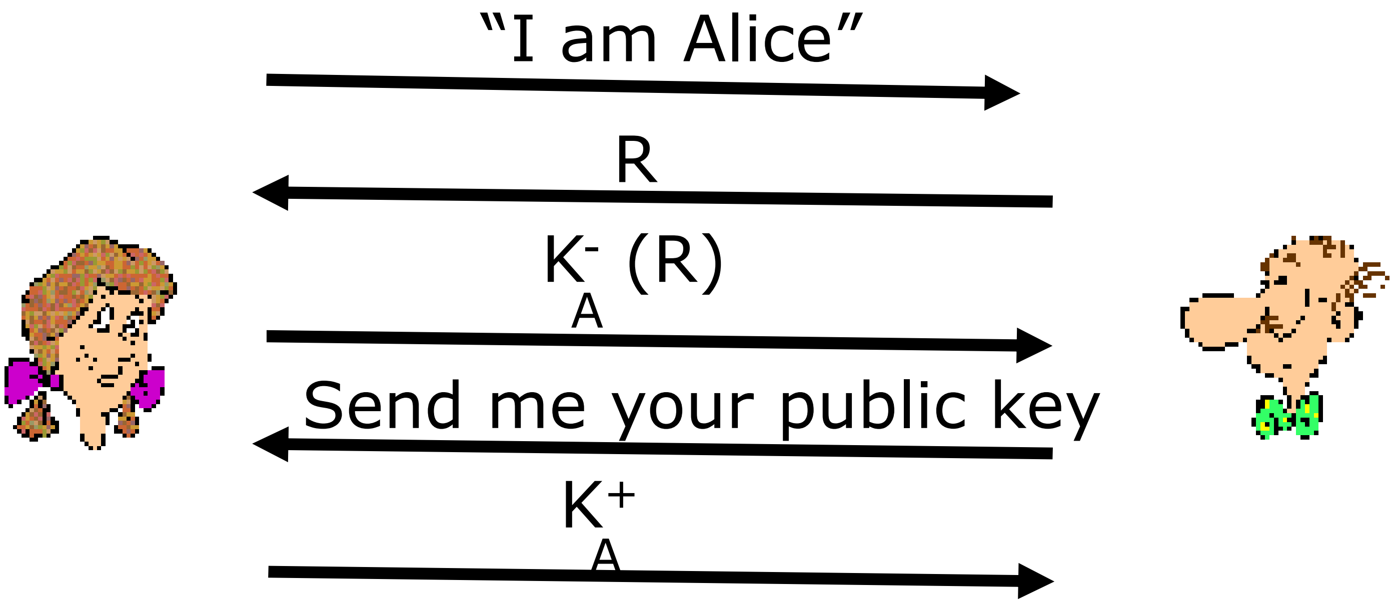


Authentication with public key

MAC requires shared symmetric key

- problem: how do Bob and Alice agree on key?
- can we authenticate using public key techniques?

Solution: use nonce, public key cryptography



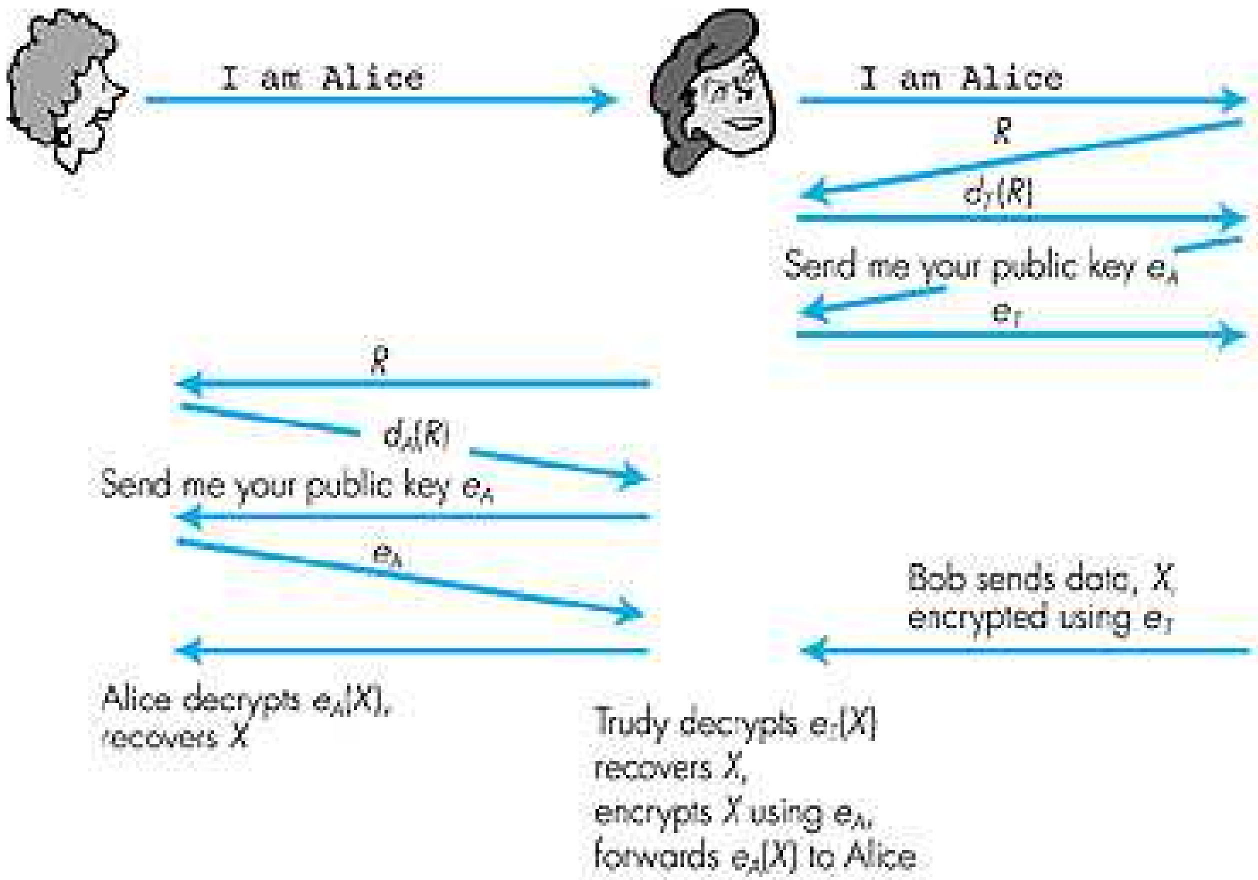
A possible security hole

PerLab

- If Bob does not require a certified public key from Alice
- Man (woman) in the middle attack
  - Trudy poses as Alice (to Bob) and as Bob (to Alice)
- Solution: always use certified public keys

ap5.0: security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



Overview

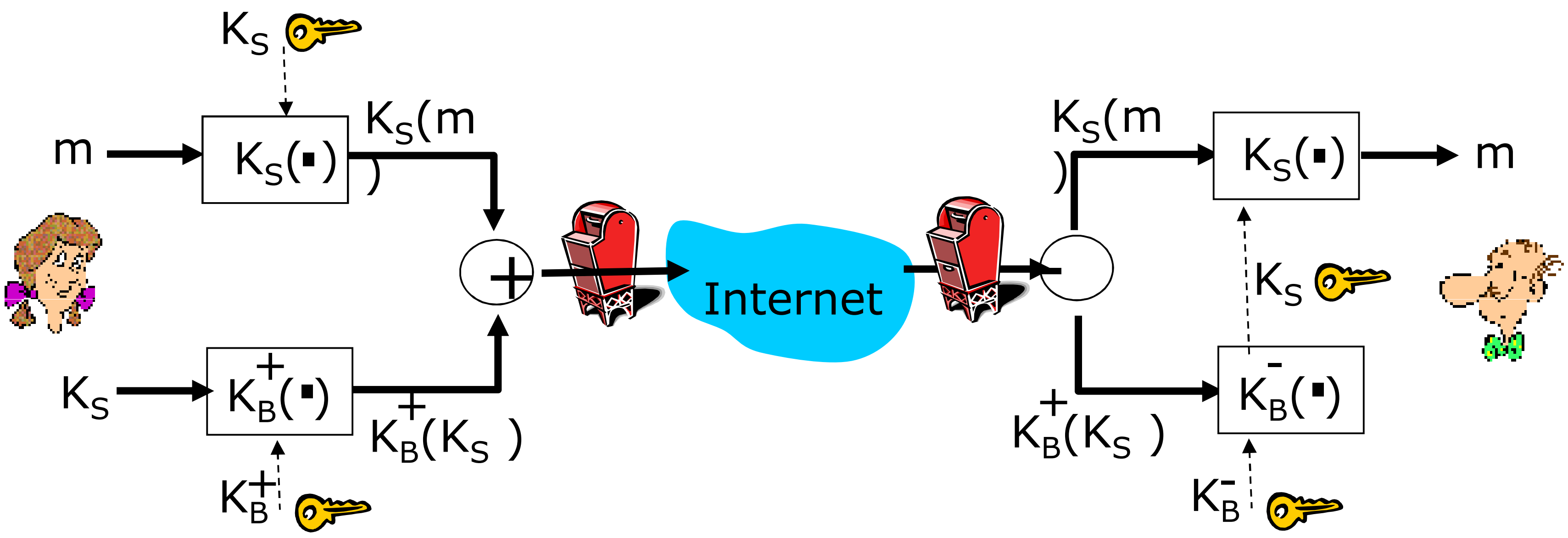
- Threats and attacks
- Cryptography as a Security Tool
    - Secrecy
    - Message integrity
    - Digital signature
    - End-to-end Authentication
    - Secure E-mail
    - Secure Socket Layer (SSL)
- Security Defenses
    - User Authentication
    - Antivirus
    - Firewalls
    - …

Secure e-mail

- Requirements
  - Confidentiality
  - Sender Authentication
  - Receiver Authentication
  - Message Integrity

Secure e-mail

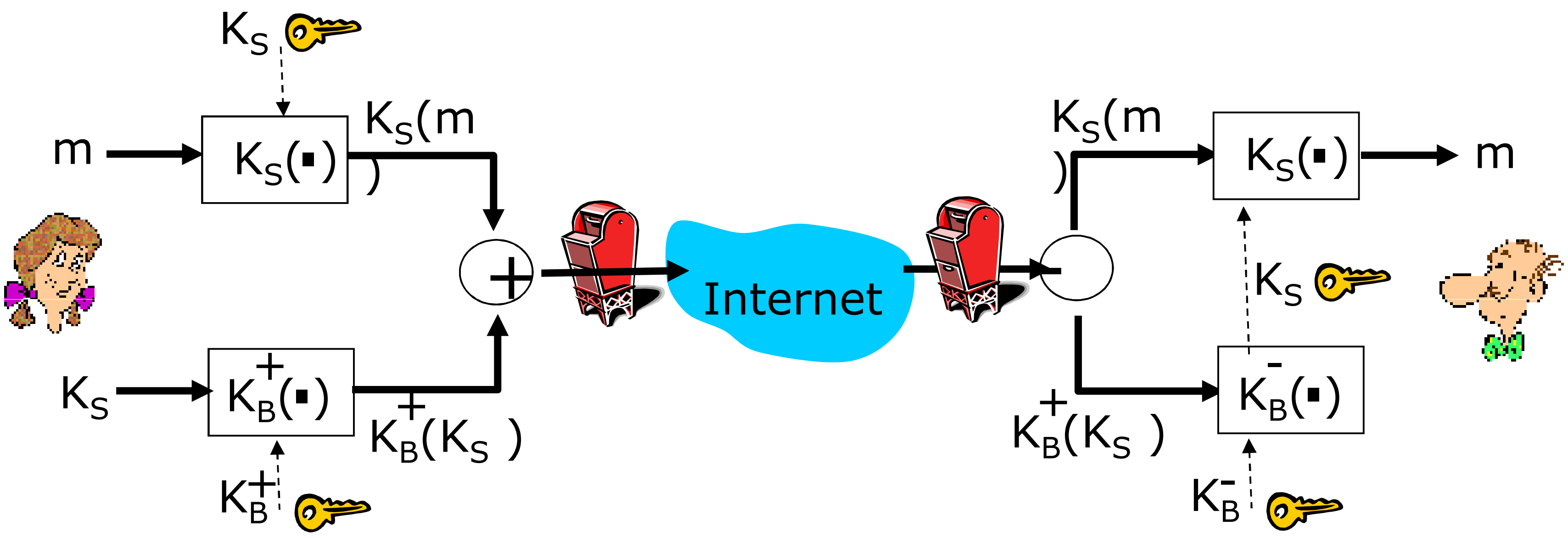
- Alice wants to send confidential e-mail, m, to Bob.



Alice:
- generates random *symmetric* private key, K_S.
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key.
- sends both $K_S(m)$ and $K_B(K_S)$ to Bob.

Secure e-mail

- Alice wants to send confidential e-mail, m, to Bob.

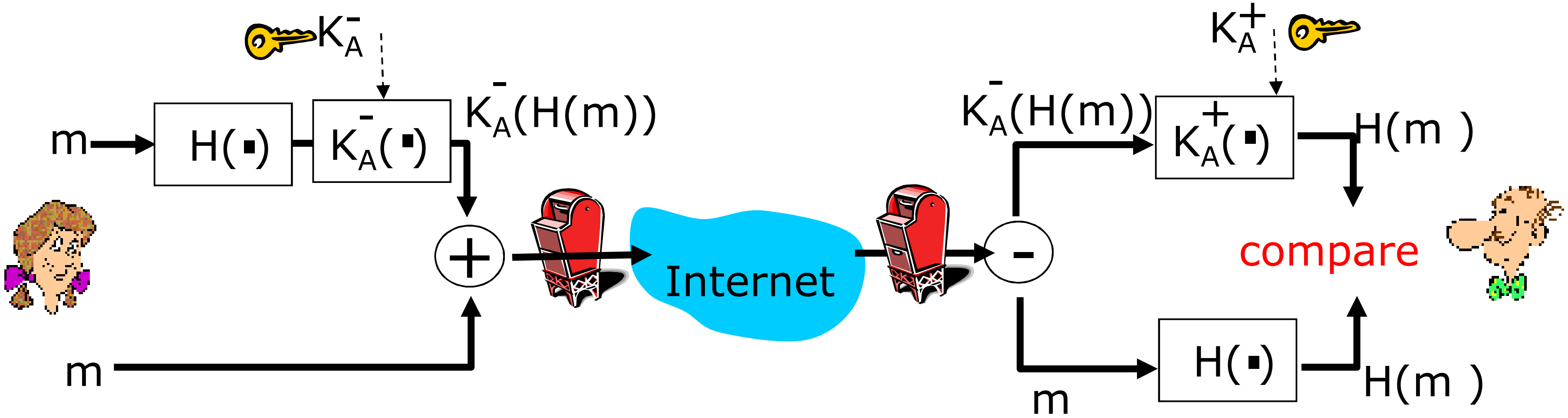


Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail (continued)

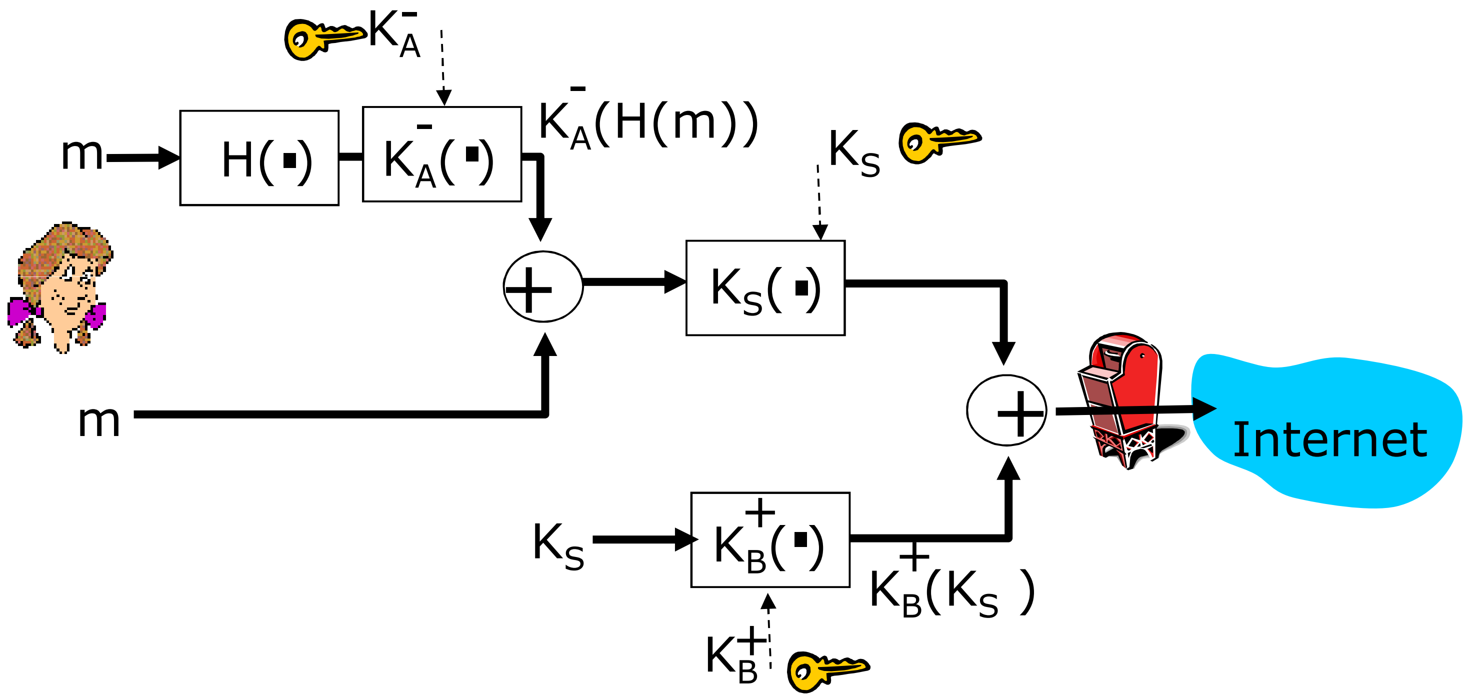
- Alice wants to provide sender authentication message integrity.



- Alice digitally signs message.
- sends both message (in the clear) and digital signature.

Secure e-mail (continued)

- Alice wants to provide secrecy, sender authentication, message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key

Pretty good privacy (PGP)

- Internet e-mail encryption scheme, a de-facto standard.
- Uses symmetric key cryptography, public key cryptography, hash function, and digital signature as described.
- Provides secrecy, sender authentication, integrity.
- Inventor, Phil Zimmerman, was target of 3-year federal investigation.

A PGP signed message:

```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

   Bob:
   My husband is out of town
   tonight. Passionately yours,
   Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJ
   hFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

Overview

- Threats and attacks
- Cryptography as a Security Tool
  - Secrecy
  - Message integrity
  - Digital signature
  - End-to-end Authentication
  - Secure E-mail
  - Secure Socket Layer (SSL)
- Security Defenses
  - User Authentication
  - Antivirus
  - Firewalls
  - …

Secure Sockets Layer (SSL)

PerLab

- PGP provides security for a specific network application
- SSL works at transport layer. Provides security to any TCP-based application using SSL services.
- Cryptographic protocol that limits two computers to only exchange messages with each other
  - Very complicated, with many variations
- Used between browsers and Web servers for secure communication (https)
  - E.g., credit card number in e-commerce applications
- SSL security services:
  - server authentication
  - data encryption
  - client authentication (optional)

SSL Encrypted Session

- Server authentication
  - The server is verified through a certificate assuring that the client is talking to correct server
- Key exchange
  - Asymmetric cryptography used to establish a secure session key (symmetric encryption) for communication
  - Browser
    - generates a symmetric session key K_s
    - encrypts it with server's public key
    - sends encrypted key to server.
  - Server
    - Using its private key, the server decrypts the session key K_s

SSL Encrypted Session

- Secure communication
  - All data sent into TCP socket (by client or server) are encrypted with session key K_s

SSL: Final Remarks

- SSL: basis of IETF Transport Layer Security (TLS).
- SSL can be used for non-Web applications, e.g., IMAP.
- Client authentication can be done with client certificates.

Overview

- Threats and attacks
- Cryptography as a Security Tool
  - Secrecy
  - Message integrity
  - Digital signature
  - End-to-end Authentication
  - Secure E-mail
  - Secure Socket Layer (SSL)
- Security Defenses
  - User Authentication
  - Antivirus
  - Firewalls
  - …

Security Defenses

- Defense in depth is most common security theory – multiple layers of security
- Security policy describes what is being secured
- Proactive Approaches
  - Access Control (User Authentication)
  - Firewall
  - Virus Protection
  - …
- Reactive Approaches
  - Auditing, accounting, and logging of all or specific system or network activities
  - Intrusion detection endeavors to detect attempted or successful intrusions

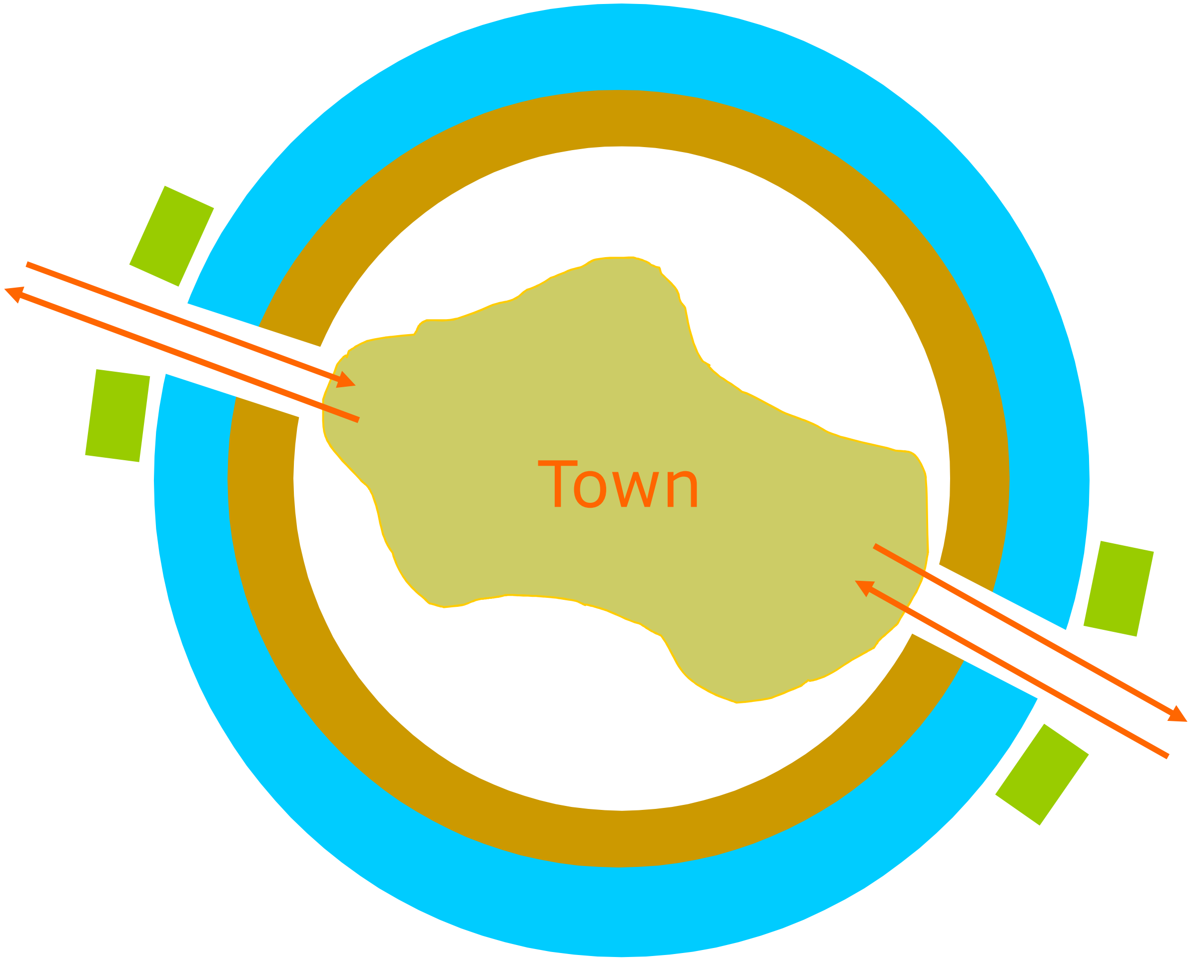
User Authentication

- Crucial to identify user correctly, as protection systems depend on user ID
- User authentication can be based on
  - Something the user *has*
    - key, card, …
  - Something the user *knows*
    - password, …
  - Something the user *is*
    - fingerprint, biometric properties, …

Passwords

- Passwords can be considered a special case of either keys or capabilities
- Passwords must be kept secret
  - Use of “non-guessable” passwords
  - Frequent change of passwords
  - Log all invalid access attempts
- Passwords may also either be encrypted or allowed to be used only once
- Good way to generate password
  - Mg’sniG!
  - My girlfriend’s name is Giulia!

Traditional Defense Principle



Lucca’s Walls

Firewall

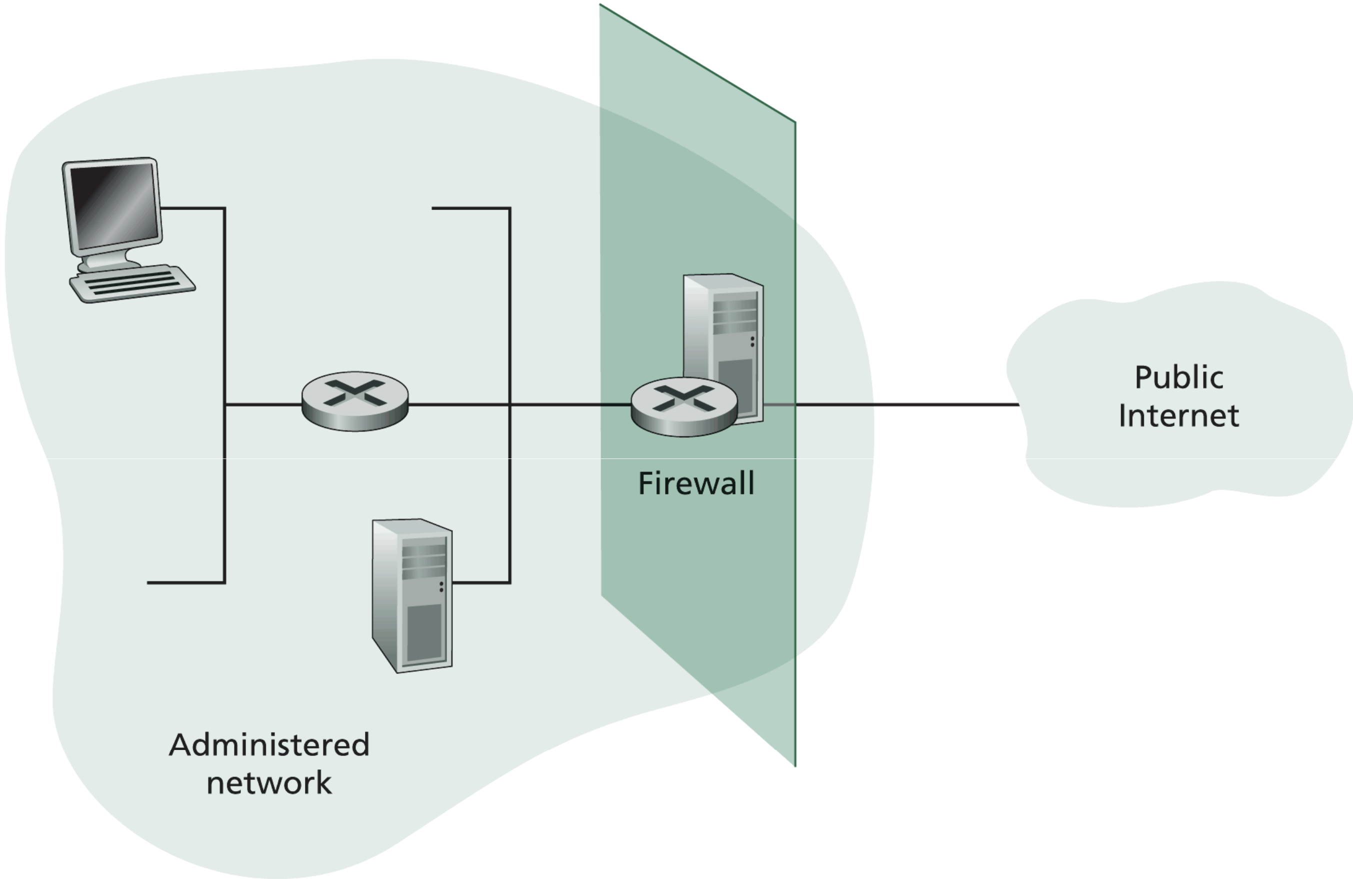
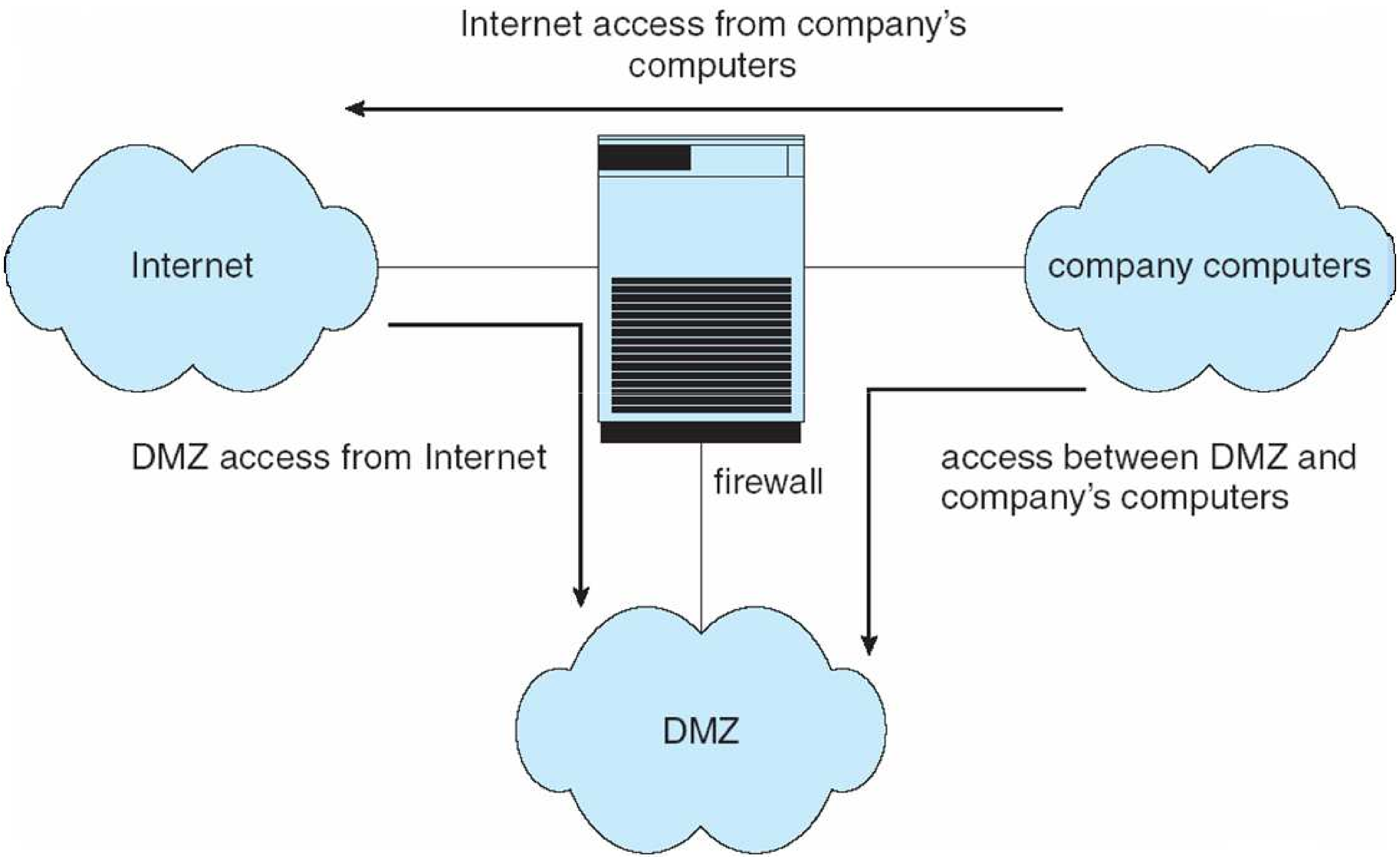


Figure 8.23 ♦ Firewall placement between the administered network and the outside world

Network Security Through Domain Separation



Firewall Classification

- A network firewall is placed between trusted and untrusted hosts
  - The firewall limits network access between these two security domains
- Personal firewall
  - Software module in our host (e.g., PC)
  - Can monitor/limit traffic to and from the host
- Packet Filtering firewall
  - permits/denies input or output of packets based on their IP addresses, port number, …
- Application Gateway
  - understands application protocol and can control them (i.e., SMTP)

Packet Filtering

- Source/Destination IP Address
- Protocol Type in IP datagrams
  - TCP, UDP, ICMP, …
- Source/Destination Port Number
- TCP flags (SYN, ACK, …)
- ICMP Message Type
- …
- Different rules for datagrams leaving/entering the internal network

Packet Filtering Rules

Rule	Source Address	Destination Address	Action	Comments
R1	111.11/16	222.22.22/24	permit	Let datagrams from Bob's university network into a restricted subnet.
R2	111.11.11/24	222.22/16	deny	Don't let traffic from Trudy's subnet into anywhere within Alice's network.
R3	0.0.0.0/0	0.0.0.0/0	deny	Don't let traffic into Alice's network.

Table 8.4 ♦ Packet-filtering rules

Packet Filtering Rules

Datagram Number	Source IP Address	Destination IP Address	Desired Action	Action Under R2, R1, R3	Action Under R1, R2, R3
P1	111.11.11.1 (hacker subnet)	222.22.6.6 (corp.net)	deny	deny (R2)	deny (R2)
P2	111.11.11.1 (hacker subnet)	222.22.22.2 (special subnet)	deny	deny (R2)	permit (R1)
P3	111.11.6.6 (univ. net, not the hacker subnet)	222.22.22.2 (special subnet)	permit	permit (R1)	permit (R1)
P4	111.11.6.6 (univ. net, not the hacker subnet)	222.22.6.6 (corp. net)	deny	deny (R3)	deny (R3)

Table 8.5 ♦ Results of packet filtering, according to rule order

Application Gateway

- Packet filtering only allows general rules
  - Deny input access to all telnet sessions (TCP port number 23)
  - Allow output access to all telnet sessions (TCP port number 23)

- Does not allow to distinguish between different users
  - E.g., Allow input access to all telnet sessions from user / IP address X
  - Possible Solution: Packet filtering router + application gateway

Application Gateway

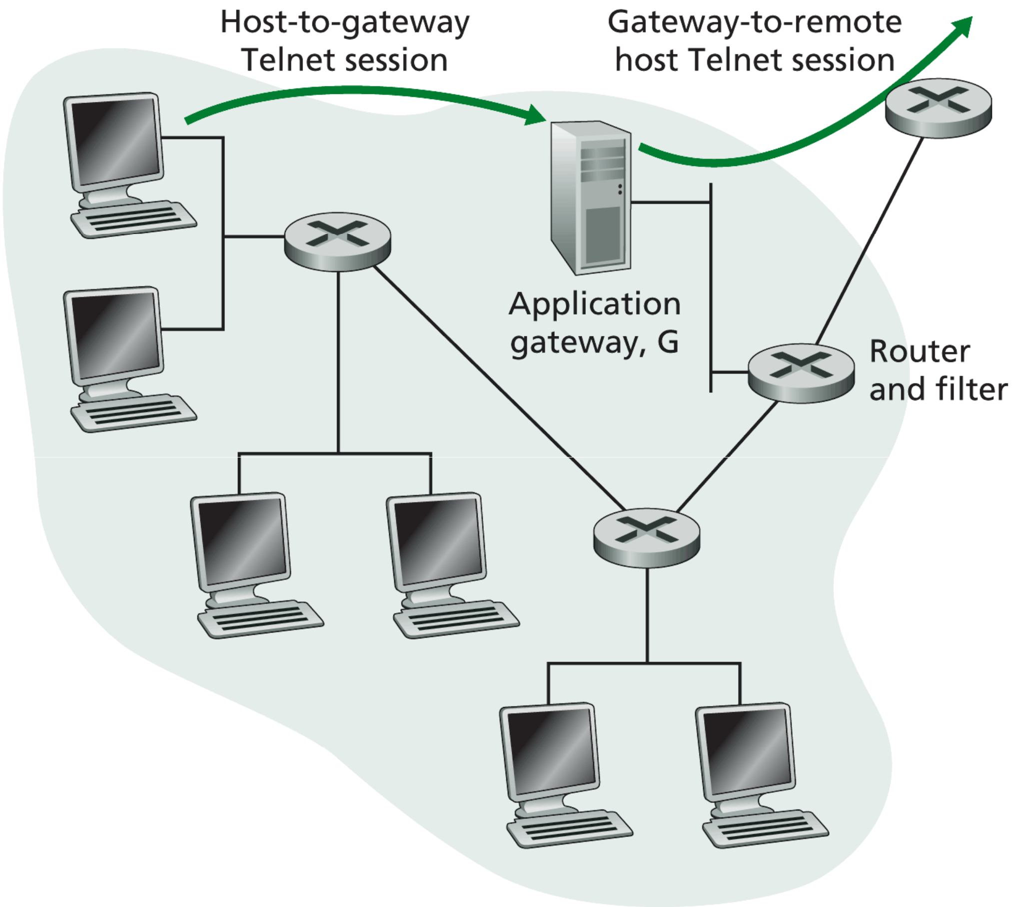


Figure 8.24 ♦ Firewall consisting of an application gateway and a filter

Application Gateway

- Limits
  - Dedicated gateway for each single application
  - Performance degradation
    - All connection must pass through the application gateway
  - The software client must be adapted to contact the application gateway

Firewall Limitations

- Can be tunneled or spoofed
  - Tunneling allows disallowed protocol to travel within allowed protocol (i.e. telnet inside of HTTP)
  - Firewall rules typically based on host name or IP address which can be spoofed
- Often use stringent policies
  - E.g., : Deny all UDP traffics
- May contains configuration bugs
  - That allows potential intruders to overcome security defenses
- May be by-passed
  - Wireless Communications
  - Communications via modem

Questions?