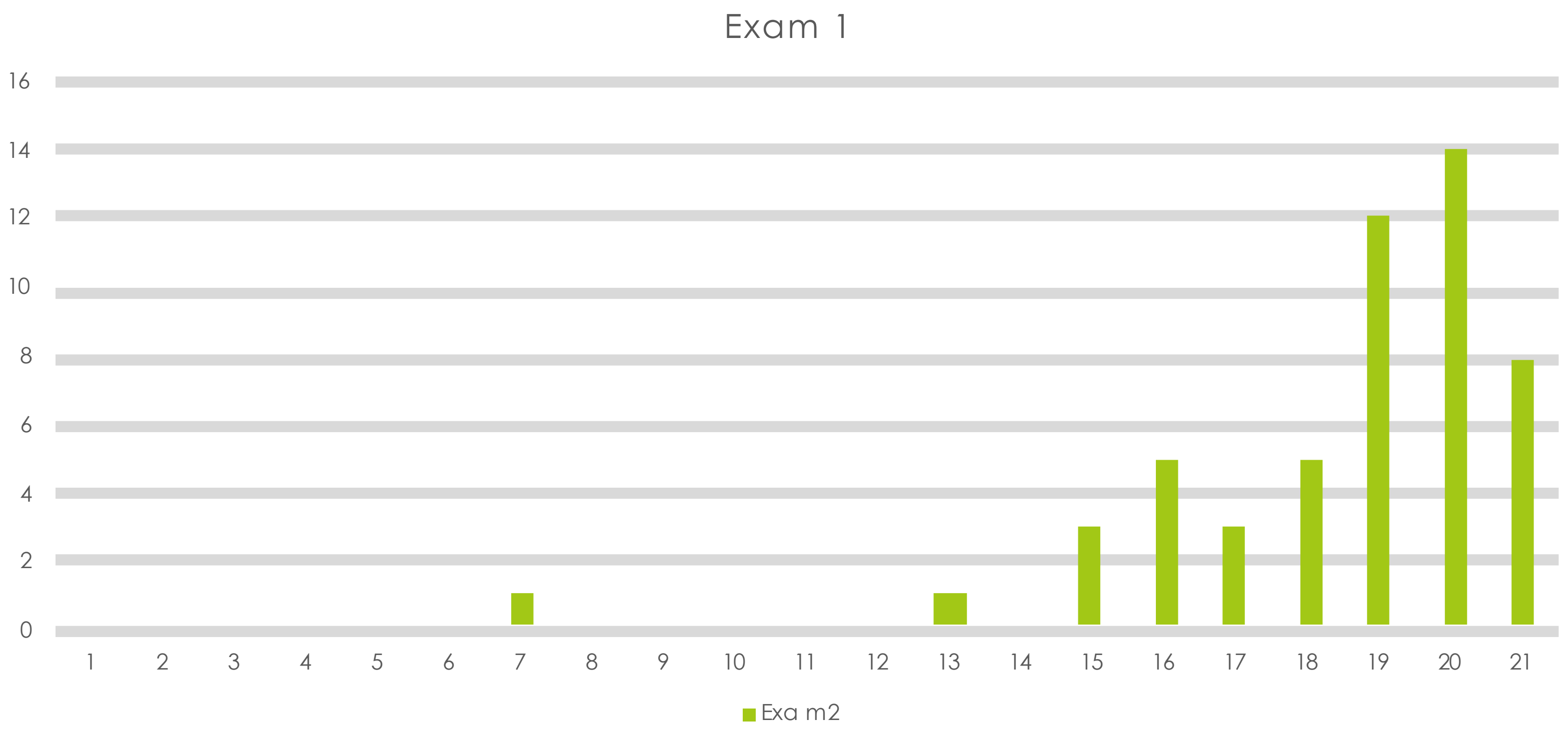
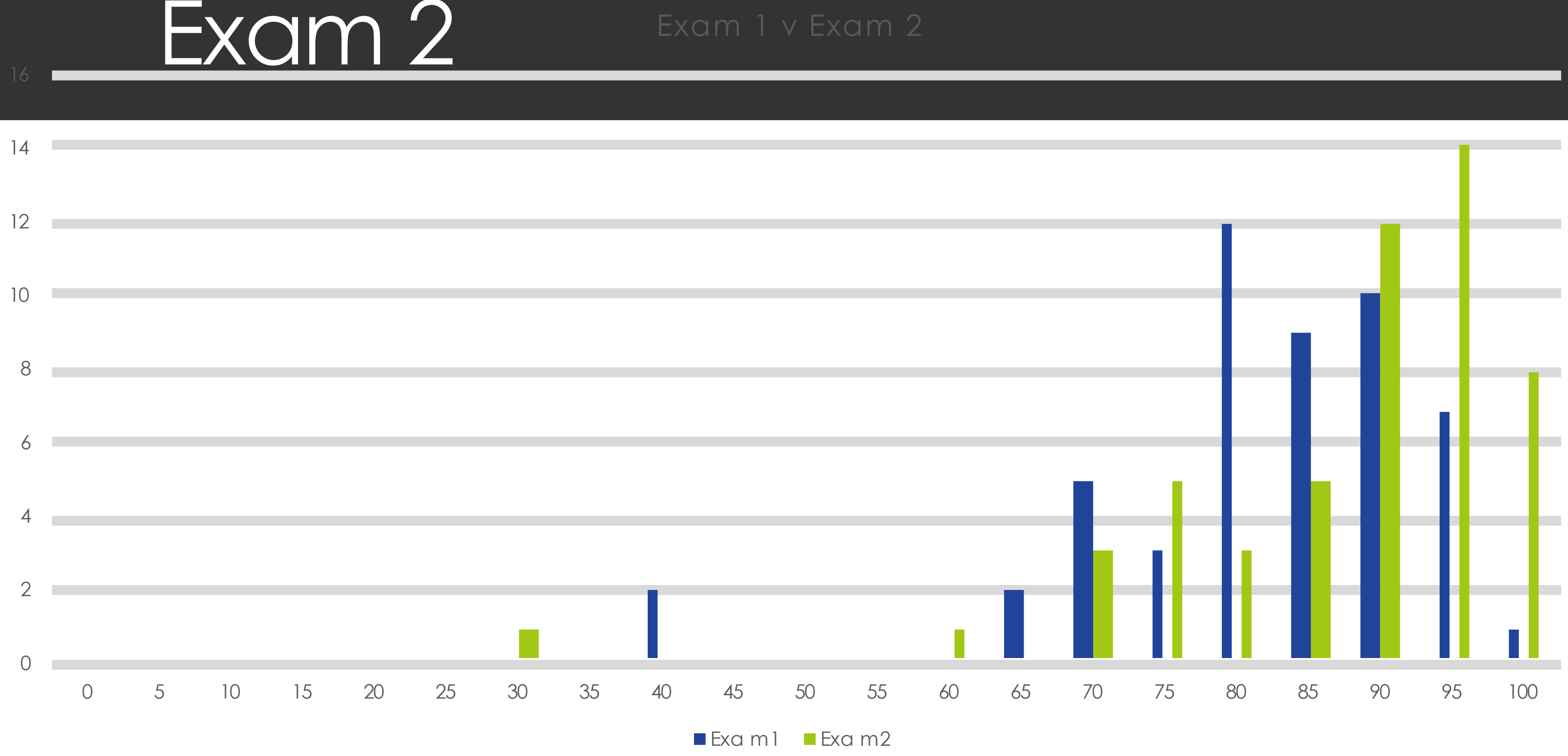
The Internet: Protocols and Security

Exam 2



Exam 1
16
14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
Exa m2

Exam 2



Exam 1 v Exam 2
16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
Exa m1
Exa m2

Announcements

- PS 10 – 11 out today
- Note: removing question from PS 10
- Monday: Lab Exam 2
- Missing Grades/Submissions?
- Monday – Thursday: Tom Cortina
- Friday: Exam 3

Lab Exam

- Bring your laptops
- 4 questions + Reference Sheet
- tkinter
  - Graphics
  - Including geometry
- 2 dimensional data collections
- Recursive functions
- Random functions

On Wednesday:

- Protocols
- History

packet switching

getting from here to there: basic transportation mechanism

The path from "here" to "there"

- For now, think of sending a message (group of bits) from one machine to another through the Internet
- We attach the source and destination IP addresses to the message
- "The Internet" gets it from source to destination
  - **but how? using packet switching**

Design Decisions

- No limit on message size
- Flexible and robust delivery mechanism

Circuit Switching
the road not taken

- Two network nodes (e.g. phones) establish a **dedicated connection** via one or more switching stations.

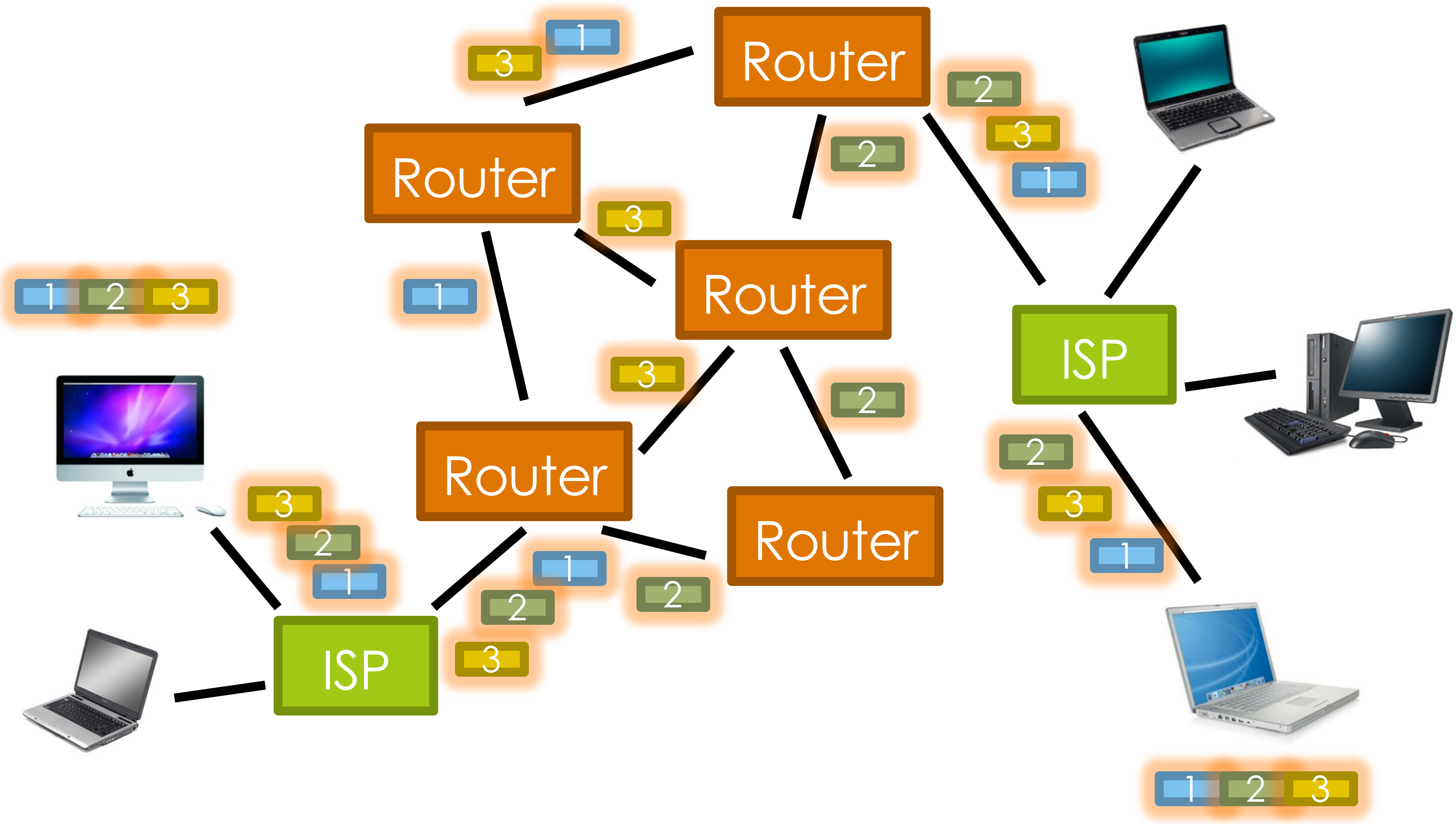
Circuit switching

- Advantages
  - reliable
  - uninterruptible
  - simple to understand
- Disadvantages
  - costly
  - inflexible
  - wasteful
  - hard to expand

Packet Switching

- Two network nodes (e.g. computers) communicate by **breaking the message up into small packets**
  - each packet sent separately
  - with a serial number and a destination address.
- *Routers* forward packets toward destination
  - table stored in router tells it which neighbor to send packet to, based on IP address of destination
- Packets may be received at the destination in any order
  - may get lost (and retransmitted)
  - serial numbers used to put packets back into order at the destination

Packet Switching



Routing and Internet structure

- Core provides transport services to edges
  - routers and gateways forward packets
  - Internet Service Providers (ISPs) provide data transmission media (fiber optic etc.)
  - domain name servers (DNS) provide directory of *host* names (more on this next time)
- Edges provide the services we humans use
  - individual users, “hosts”
  - private networks (corporate, educational, government…)
  - business, government, nonprofit services

end-to-end principle

Internet article of faith

Core architectural guideline

- Idea: *routers should stick to getting data quickly from its source to its destination!*
  - they can be fast and stupid
- Everything else is responsibility of edges, *e.g.*
  - error detection and recovery
  - confidentiality via encryption
  - …

Benefits of End-to-end

- Speed and flexibility
- Support for innovation: routers need know nothing about apps using their services
- Equality of uses: routers can't discriminate based on type of communication (*net neutrality*)

Governing the Internet

- Internet Society: a range of partners from non-profit agencies, local and global NGOs, academia, technologists, local councils, federal policy and decision makers, business (www.isoc.org)

- Internet Service Providers (ISPs) regulated in the USA by the Federal Communications Commission (FCC)

The Internet and Python

Sending email

```
# mail (run where there is a local mail server)

import smtplib
from email.mime.text import MIMEText

def mail_demo() :
    msg = MIMEText('Give me an A!')
    msg['Subject'] = 'My grade'
    msg['From'] = 'student@example.org'
    msg['To'] = ’jmfrye@andrew.cmu.edu'
    server = smtplib.SMTP('localhost')
    server.send_message(msg)
    server.quit()
```

Fetching a web page

```
# web (run this wherever)

from urllib.request import urlopen

def web_demo() :
    page = urlopen('http://www.cs.cmu.edu/~15110')
    print("Opened URL ", page.geturl())
    print("Contents:")
    for line in page :
        print(line.decode('ISO-8859-1'))
```

Higher Protocols

“Higher” and “lower” level protocols

- Network protocols are organized in *layers*
  - IP packet delivery is the lowest *layer* of the Internet protocol *stack*
  - “Higher” layers use services provided by “lower” layers
  - Each layer is responsible for a type of service

Layers of the Internet ("higher" to "lower")

- Application Layer provides services to human beings
  - *e.g.* browser, email client, Skype
- Transport Layer provides services to applications
  - converts between application messages and IP packets
  - figures out which application to deliver a message to
  - possibly detects and corrects delivery errors
- Internet Layer provides services to transport layer
  - determines next "hop" for a packet and sends it there
- Link Layer provides services to internet layer
  - physically converts between signals and bits

Example: Layering the Web

CLIENT MACHINE		SERVER MACHINE
ask for a web page	HTTP	send a web page
request connection	TCP	acknowledge request
best-effort packets	IP	best-effort packets
physical data transport	(various)	physical data transport

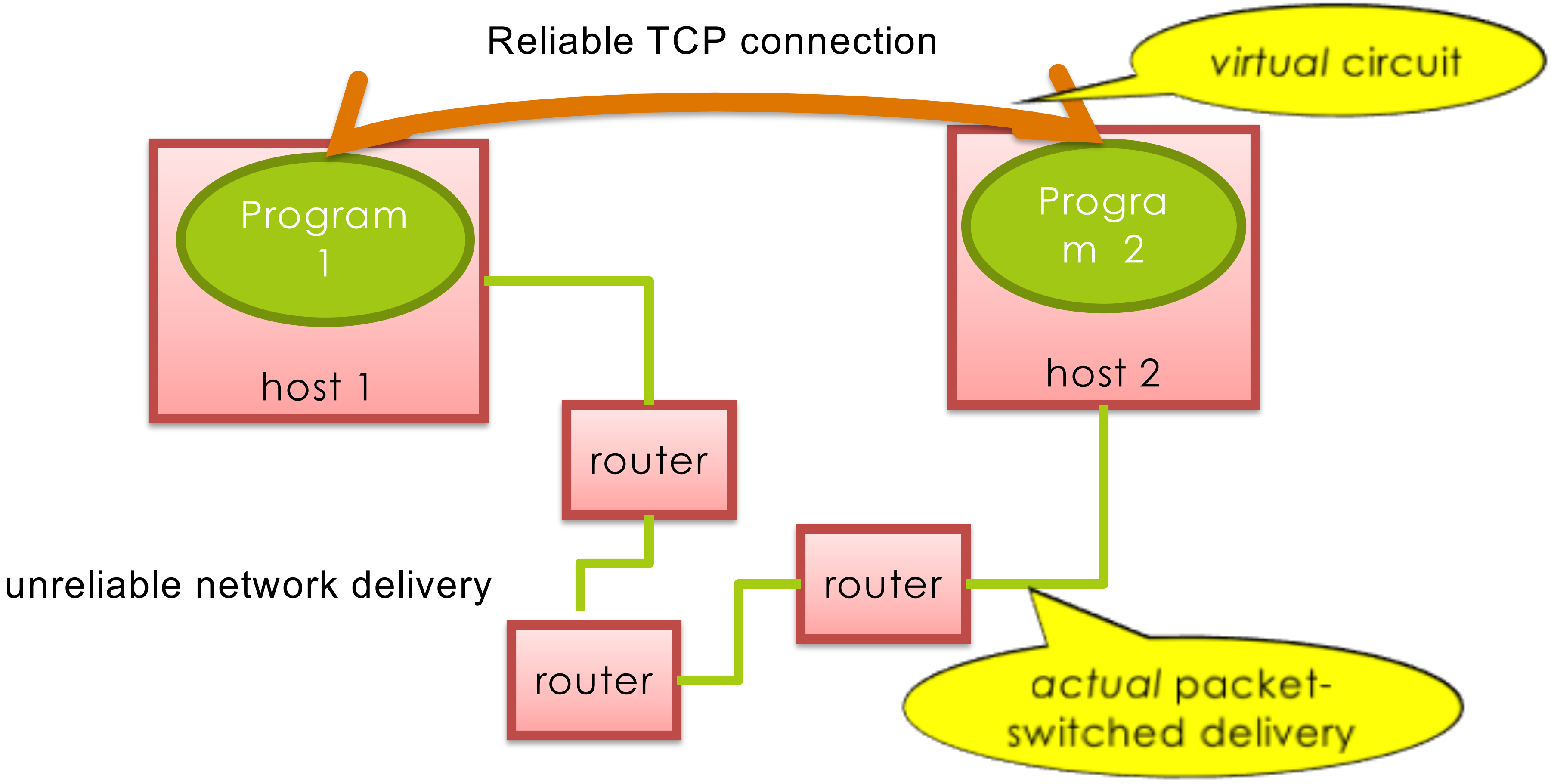
Transport Layer

from IP packets to application messages

Transport Layer

- Splits application messages into IP packets and maps applications to *port number*
  - IP address identifies machine, but port number identifies an application operating on that machine (web, email, etc.)
- Transport Control Protocol (TCP)
  - Creates a *reliable* bi-directional stream (source address/port and destination address/port)
- User Datagram Protocol (UDP)
  - Creates a single one-way message to a remote application (destination address/port)
    - used for voice, video, DNS lookup, …

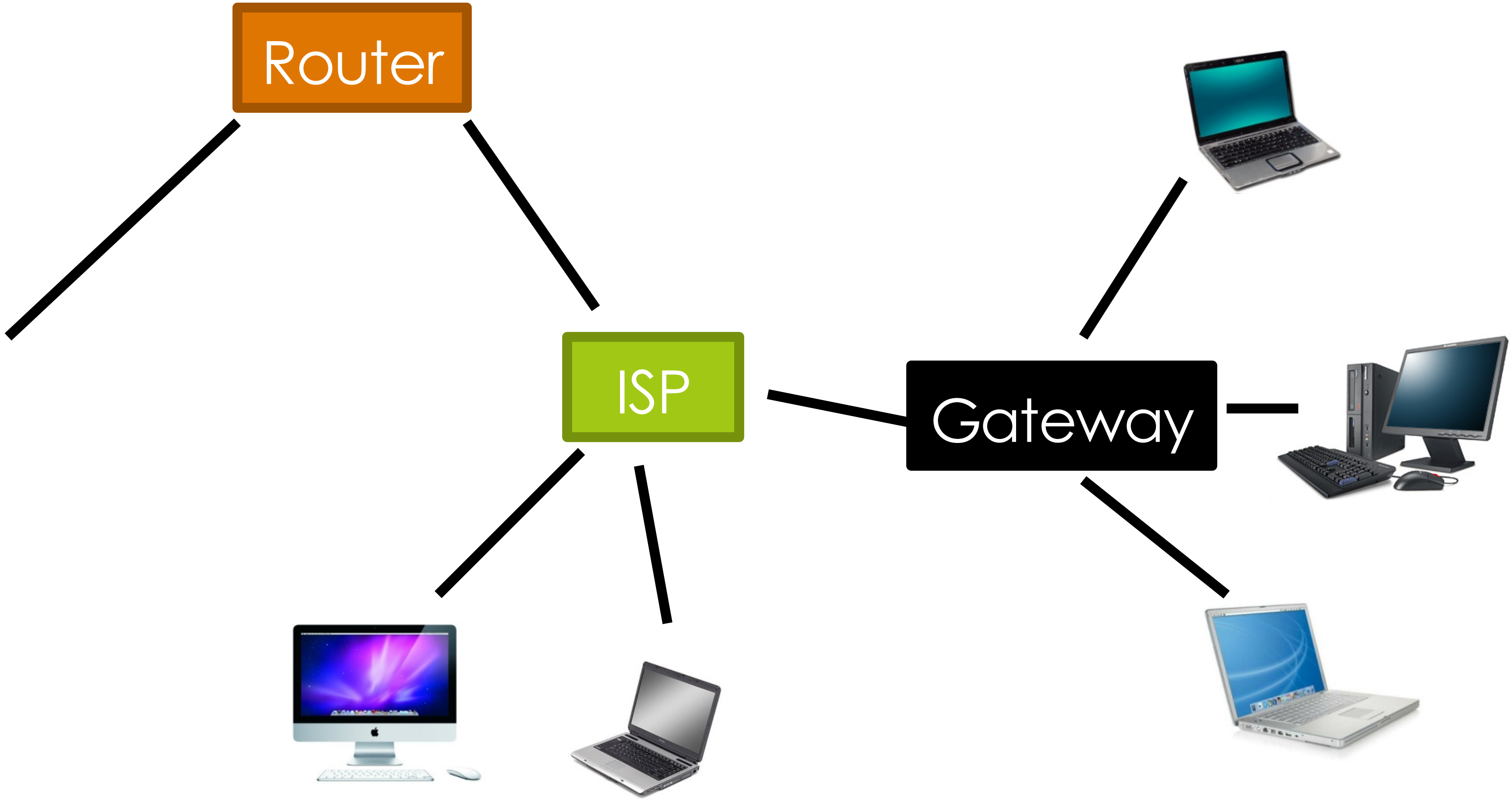
Transport Layer



Reliable Communication with TCP

- Suppose A and B are the TCP programs of two computers.
  - An application asks A to send a message to an application at B.
  - A breaks the message into several packets.
    - Each packet includes parity information, so B can check it for accuracy.
    - Packets are sent via IP.
  - B receives the packets.
    - If B is missing a packet or receives a corrupt packet, it can request retransmission.
    - If the packet is OK, B sends an acknowledgement.
  - If A doesn't get an acknowledgement, it will retransmit.
  - B assembles the incoming packets in order and provides the message to the appropriate application.

Network Address Translation (NAT)



Network Address Translation (NAT)

- Used to accommodate more users on the Internet, security, and administration.
- The gateway assigns an additional code called a port for each user. Packets are tagged with the port.
- The gateway knows where to route the messages on the private network, but all messages from that private network share the same single IP address.

Domain names

from **98.139.183.24** to **yahoo.com**

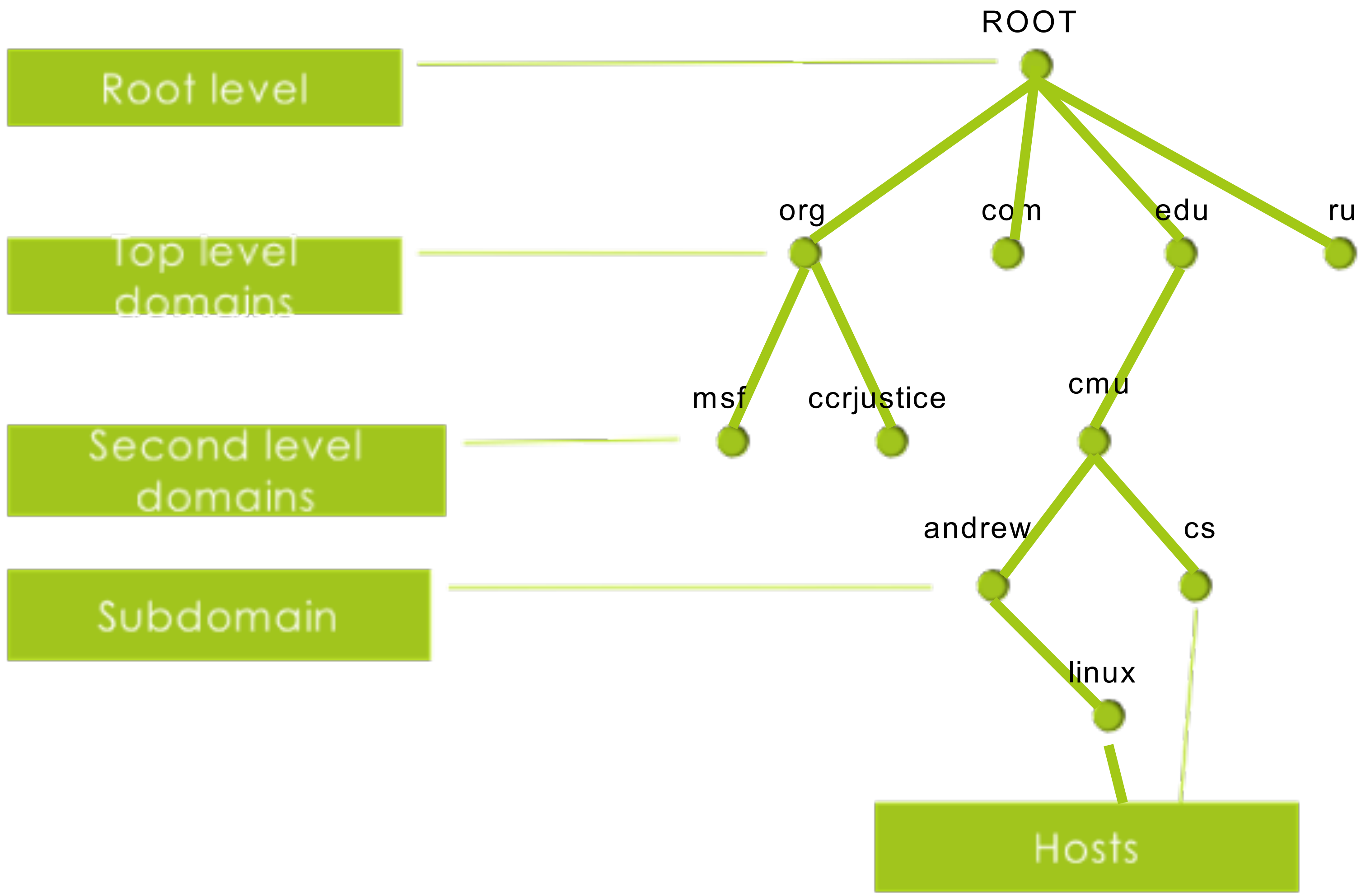
From names to IP addresses

- URL: http://www.andrew.cmu.edu/user/nbier/15110/index.html
- Email address: nbier@andrew.cmu.edu
- We don’t want IP addresses in our URLs or email addresses—why not?
- Domain Name Service (DNS) *translates* names to addresses

DNS design

- Problem: so many names! How to make lookup fast?
- Solution: hierarchy of name servers
  - Each machine knows a name server, which knows how to find a root name server
  - **root** name servers know DNS servers for each top-level domain (e.g., "edu”, “com”, “net”, “uk”, “ru”)
  - **top-level** domain servers know DNS servers for each second-level domain (e.g., "cmu.edu”, “co.uk”)
  - **second-level** domain servers know **each host** directly in their domain (e.g., "www.cmu.edu") and DNS servers for each **third-level** domain (e.g., "andrew.cmu.edu")

DNS Hierarchy (fragment)



DNS Lookup

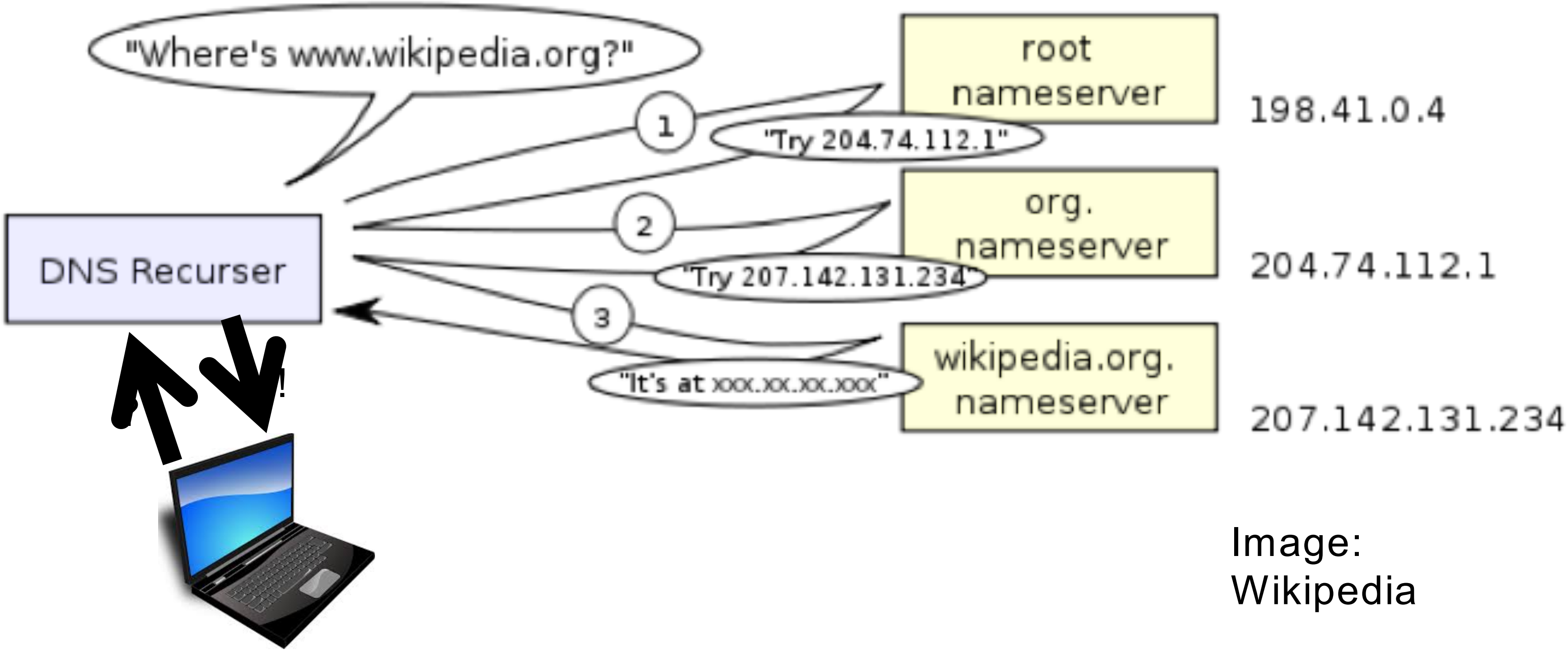
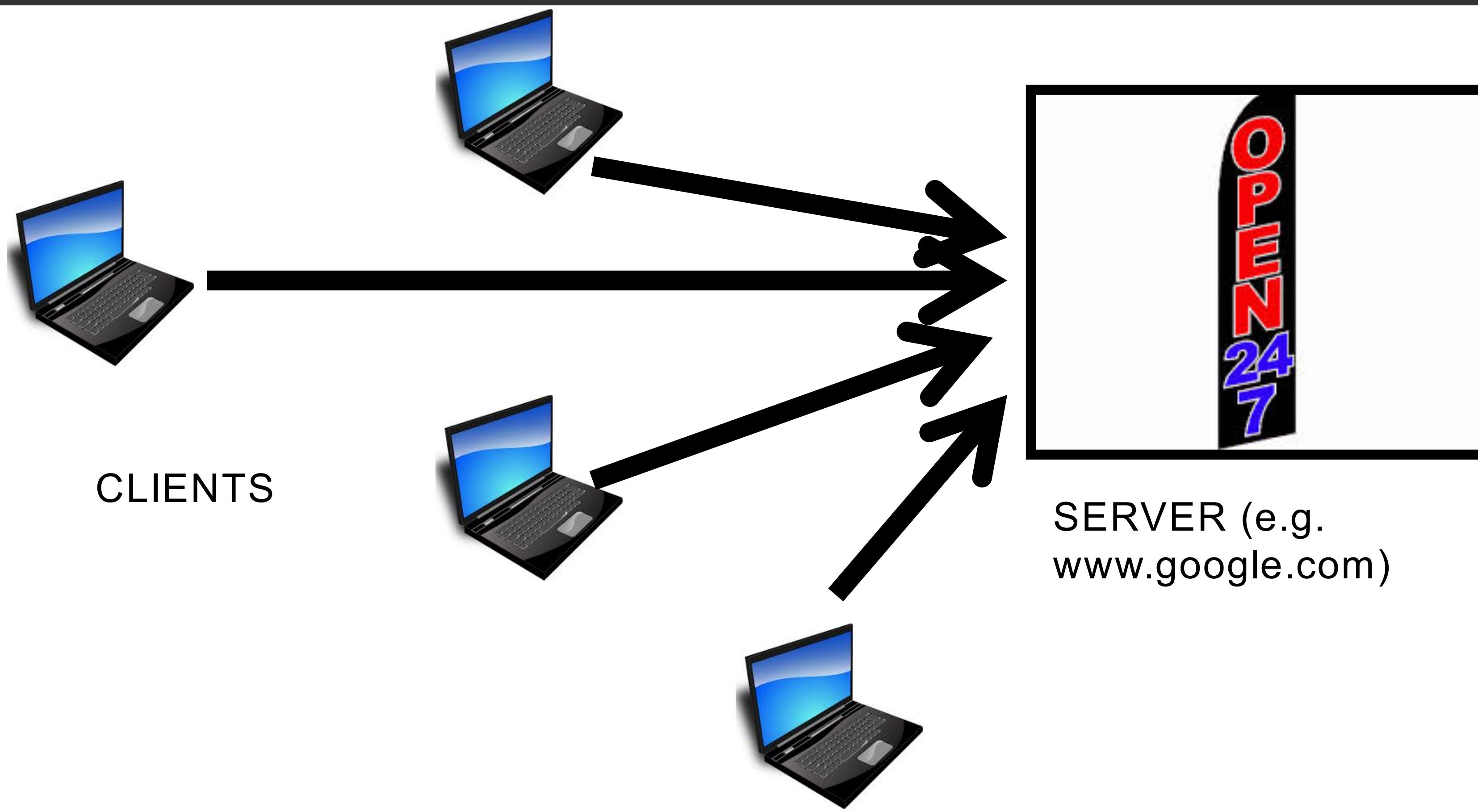


Image: Wikipedia

Client-server architectures

web, mail, streaming video, and more

Client-server Architectures



Client-server Architectures

- Architecture: an organizing principle for a computing system
- Most common architecture for Internet applications: *client-server*
- Server is always on, waiting for requests
  - *server software* (e.g. Apache) tells TCP (transport layer software) on its own machine "please listen for messages with port number 80"
  - *client software* (e.g. Chrome) tells TCP "please send this message to machine xxx.xxx.xxx.xxx with port number 80"
  - TCP gives message to IP, which sends it through internet to server machine; IP at server machine delivers to TCP at server machine
  - TCP at the server machine delivers the message to Apache

The Web

- World Wide Web = html + http

- html = HyperText Markup Language, an encoding
  - tells what a page should look like and
  - what other pages it links to

- http = HyperText Transfer Protocol
  - agreement on how client and server interact

HTML: an encoding

- Example: using your favorite plain-text editor create the following text file:

```
<html><head>
<title>15110, Summer '17,
Example web page</title>
</head>
<body>
<h1>Hello World!</h1>
</body></html>
```

Nothing to do with the Internet!

- In a browser type its name in the address bar, e.g.

```
file:///Users/pennyanderson/CMU/110/week11/example1.html
```

HTML: networked hypertext

- Now add

```
<a href=http://en.wikipedia.org/wiki/Hello_world_program>
Hello World!</a>
```

Code for getting information across the Internet

- save as example2.html

  and load

HTTP: hypertext transfer protocol

- Protocol for communication between web client *application* (e.g. Chrome, Safare, IE, Firefox) and web server *application* (e.g. Apache)

- Agreement on how to ask for a web page, how to send data entered into a form, how to report errors (codes like *404 not found*), etc.

Uniform Resource Locators

- A Web page is identified by a Uniform Resource Locator (URL)

  protocol://host address/page

- A URL

http://www.cs.cmu.edu/~15110/index.html

Protocol to use

Overview of web page delivery

1. Web browser (client) translates name of the server to an IP address (e.g. 128.2.217.13) (using DNS)
2. Establishes a TCP connection to 128.2.217.13 port 80
3. Constructs a message

   GET /~15110/index.html HTTP/1.1

4. Sends the message using TCP/IP
5. Web server locates the page and sends it using services of TCP/IP
6. The connection is terminated

Layers and Encapsulation

Separate responsibilities

- Message:“GET /~15110/index.html HTTP/1.1”

  Request/get web page

- TCP segment:
  control information including sequence number, so-called port number for web server;
  + message

  Connect client and server reliably

- IP packet:
  control info including source address, destination address, fragment sequencing information + TCP segment

  Best-effort packet switching

Summary

- Applications communicate on the Internet via *application protocols* like
  - HTTP for the web
  - SMTP for email
  - RTSP for streaming media
- Application protocols rely on
  - Domain Name Servers for name translation, and
  - *transport protocols* like
    - TCP for reliable two-way connections
    - UDP for one-way “datagrams”
- Transport protocols rely on IP for packet delivery

Security issues

Networking is a security issue

- Why?
- If you want a really secure machine, lock it in an electromagnetically shielded room and don’t connect it to any networks or other sources of data beyond your control.
- Not much fun, is it?

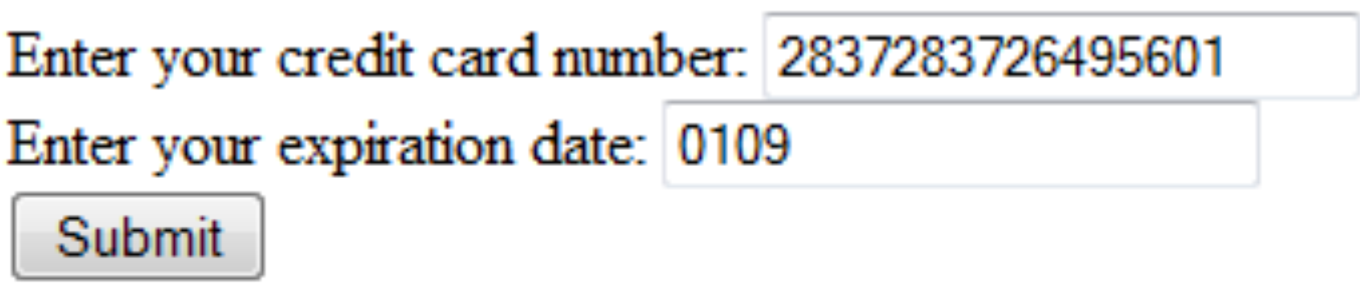
The Problem

- The Internet is public
  - Messages sent pass through many machines and media
- Anyone intercepting a message might
  - read it and/or
  - replace it with a different message
- The Internet is anonymous
  - IP addresses don't establish identity
- Anyone may send messages under a false identity

Cryptography offers *partial* solutions to all of these problems

A Shady Example

- I want to make a purchase online and click a link that takes me to http://www.sketchystore.com/checkout.jsp
- What I see in my browser:



A Shady Example (cont'd)

- When I press SUBMIT, my browser sends this:

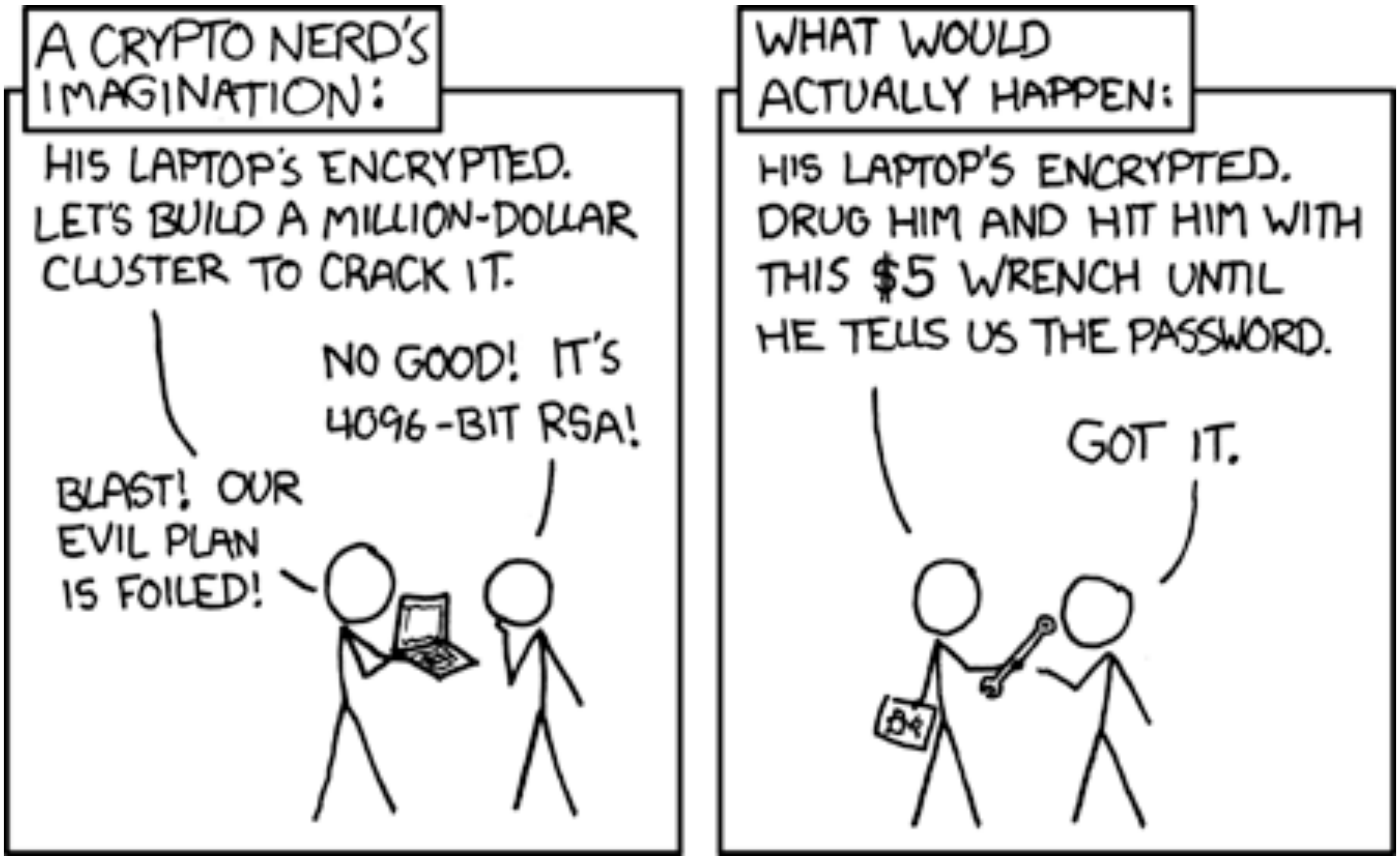
```
POST /purchase.jsp HTTP/1.1
Host: www.sketchystore.com
User-Agent: Mozilla/4.0
Content-Length: 48
Content-Type: application/x-www-form-urlencoded
userid=rbd&creditcard=2837283726495601&
   exp=01/09
```

A Shady Example (cont’d)

- If this information is sent unencrypted, who has access to my credit card number?
  - Other people who can connect to my wireless ethernet
  - Other people physically connected to my wired ethernet
  - …
- Packets are passed from router to router.
  - *All* those routers have access to my data.

A caveat

cryptography is not security


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

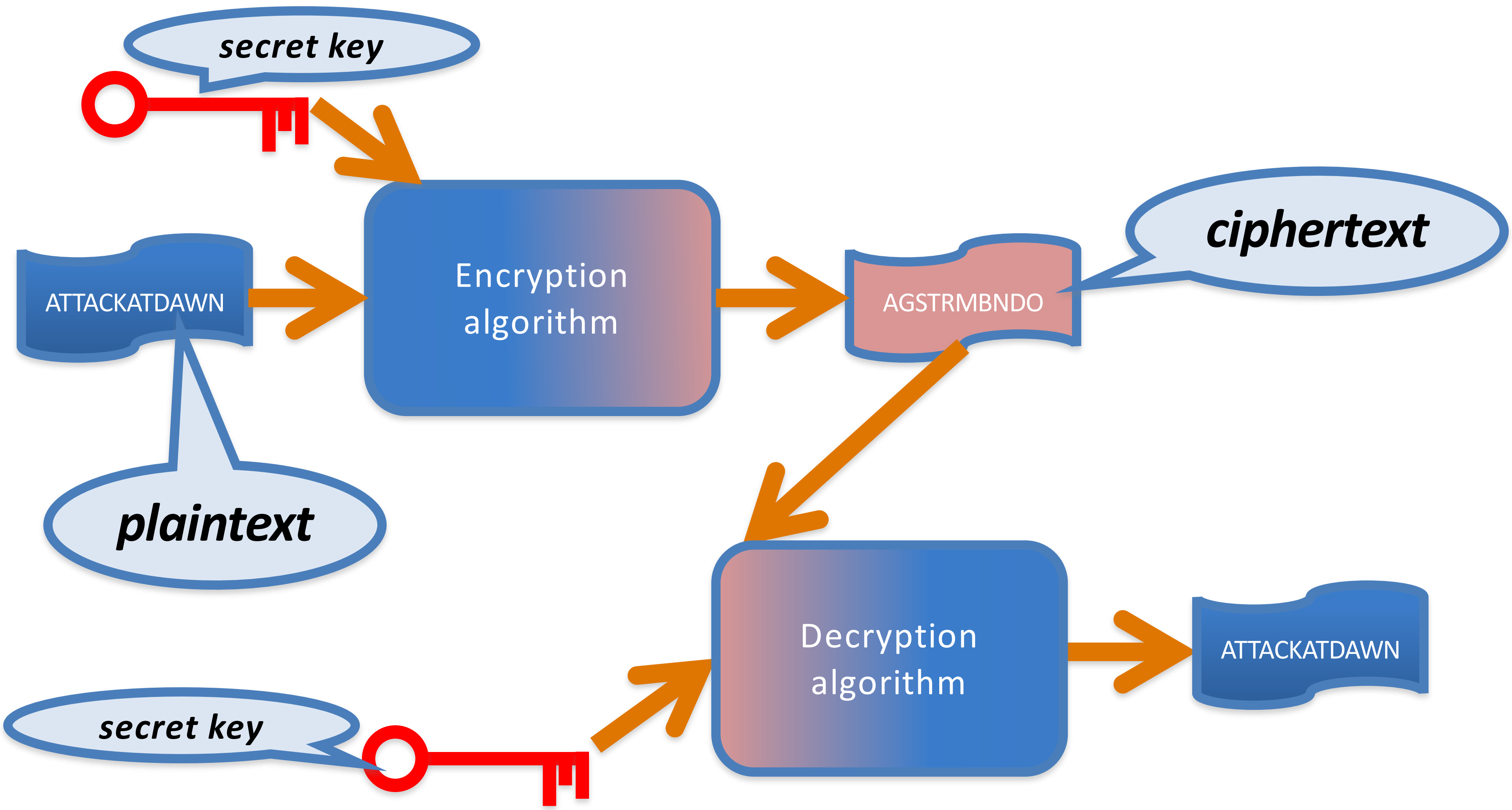
Encryption and cryptanalysis

basic concepts

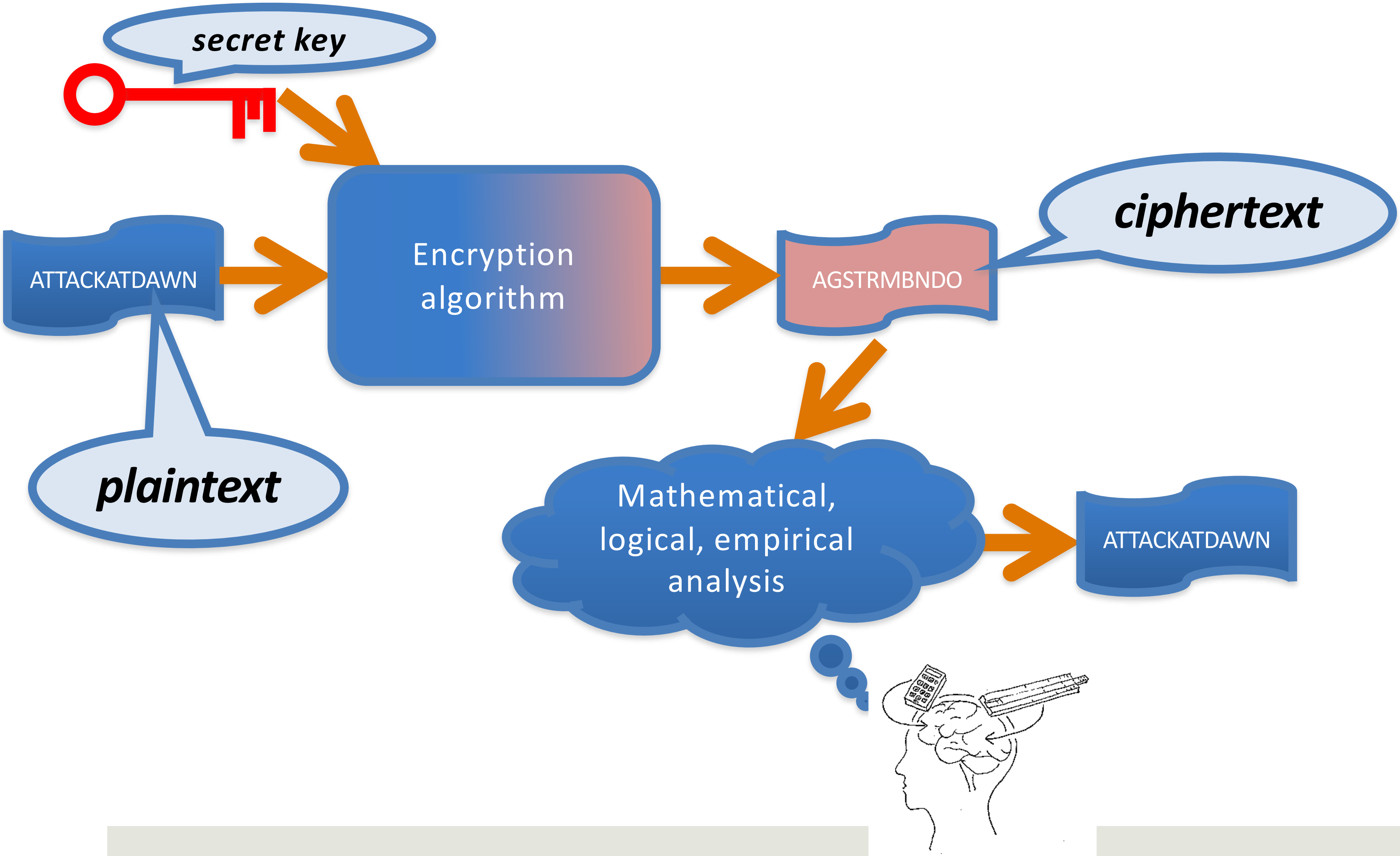
Encryption

- We encrypt (encode) our data so others can’t understand it (easily) except for the person who is supposed to receive it.
- We call the data to encode plaintext and the encoded data the ciphertext.
- Encoding and decoding are *inverse functions* of each other

Encryption/decryption



Cryptanalysis



Encryption techniques

substitution and transposition

Two basic ways of altering text to encrypt/decrypt

- Substitute one letter for another using some kind of rule

Substitution cipher

- Scramble the order of the letters using some kind of rule

Transposition cipher

Substitution Ciphers

- Simple encryption scheme using a substitution cipher:
  - Shift every letter forward by 1:

    A $\longrightarrow$ B, B $\longrightarrow$ C, ..., Z $\longrightarrow$ A

- Example:
  MESSAGE $\longrightarrow$ NFTTBHF

- Can you decrypt TFDSFU?

Substitution Ciphers

- Simple encryption scheme using a substitution cipher:
  - Shift every letter forward by 1:

    A → B, B → C, ..., Z → A
- Example:
  MESSAGE → NFTTBHF
- Can you decrypt TFDSFU? SECRET

Caesar Cipher

- Shift forward *n* letters; *n is the secret key*
- For example, shift forward 3 letters:
  A ⟶ D, B ⟶ E, ..., Z ⟶ C
  - This is a Caesar cipher using a **key** of 3.
- MESSAGE ⟶ PHVVDJH
- How can we crack this encrypted message if we don't know the key?
  DEEDUSEKBTFEIIYRBOTUSETUJXYI

Caesar Cipher (cont'd)

DEEDUSEKBTFEIIYRBOTUSETUJXYI
EFFEVTFLCUGFJJZSCPUVTFUVKYZJ

FGGFWUGMDVHGKKATDQVWUGVWLZAK
GHHGXVHNEWIHLLBUERWXVHWXMABL

HIIHYWIOFXJIMMCVFSXYWIXYNBCM
IJJIZXJPGYKJNNDWGTYZXJYZOCDN

JKKJAYKQHZLKOOEXHUZAYKZAPDEO
KLLKBZLRIAMLPPFYIVABZLABQEFP

LMMLCAMSJBNMQQGZJWBCAMBCRFGQ
MNNMDBNTKCONRRHAKXCDBNCDSGHR

NOONECOULDPOSSIBLYDECODETHIS
OPPOFDPVMEQPTTJCMZEFDPEFUIJT
PQQPGEQWNFRQUUKDNAFGEQFGVJKU

QRRQHFRXOGSRVVLEOBGHFRGHWKLV
RSSRIGSYPHTSWWMFPCHIGSHIXLMW

STTSJHTZQIUTXXNGQDIJHTIJYMNX
TUUTKIUARJVUYYOHREJKIUJKZNOY

UVVULJVBSKWVZZPISFKLJVKLAOPZ
VWWVMKWCTLXWAAQJTGLMKWLMBPQA

WXXWNLXDUMYXBBRKUHMNLXMNCQRB
XYYXOMYEVNZYCCSLVINOMYNODRSC

YZZYPNZFWOAZDDTMWJOPNZOPESTD
ZAAZQOAGXPBAEEUNXKPQOAPQFTUE

ABBARPBHYQCBFFVOYLQRPBQRGUVF

BCCBSQCIZRDCGGWPZMRSQCRSHVWG
CDDCTRDJASEDHHXQANSTRDSTIWXH

- How long would it take a computer to try all 25 shifts?

Vigenère Cipher

- Shift different amount for each letter. Use a *key word*; each letter in the key determines how many shifts we do for the corresponding letter in the message.
- Example: key word "`cmu`": shift by 2, 12, 20
- Message "`pittsburgh`"

`cmucmucmuc`

encrypted: `runvevwdaj`

- Try it yourself at http://www.simonsingh.net/The_Black_Chamber/v_square.html

ABCDEFGHIJKLMNOPQRSTUVWXYZ

A	ABCDEFGHIJKLMNOPQRSTUVWXYZ	no shift
B	BCDEFGHIJKLMNOPQRSTUVWXYZA	shift by 1
C	CDEFGHIJKLMNOPQRSTUVWXYZAB	shift by 2
D	DEFGHIJKLMNOPQRSTUVWXYZABC	shift by 3
E	EFGHIJKLMNOPQRSTUVWXYZABCD	etc.
F	FGHIJKLMNOPQRSTUVWXYZABCDE	

...

- Message: ATTACKATDAWN
- Pick a secret key DECAFDECAFDE
- Encrypted: D

1st letter in the message is shifted by 3, 2nd letter is shifted by 4, ...

	ABCDEFGHIJKLMNOPQRSTUVWXYZ
A	ABCDEFGHIJKLMNOPQRSTUVWXYZ
B	BCDEFGHIJKLMNOPQRSTUVWXYZA
C	CDEFGHIJKLMNOPQRSTUVWXYZAB
D	DEFGHIJKLMNOPQRSTUVWXYZABC
E	EFGHIJKLMNOPQRSTUVWXYZABCD
F	FGHIJKLMNOPQRSTUVWXYZABCDE

...

- Message: ATTACKATDAWN
- Pick a secret key DECAFDECAFDE
- Encrypted: DX

1st letter in the message is shifted by 3, 2nd letter is shifted by 4, …

ABCDEFGHIJKLMNOPQRSTUVWXYZ

A ABCDEFGHIJKLMNOPQRSTUVWXYZ

B BCDEFGHIJKLMNOPQRSTUVWXYZA

C CDEFGHIJKLMNOPQRSTUVWXYZAB

D DEFGHIJKLMNOPQRSTUVWXYZABC

E EFGHIJKLMNOPQRSTUVWXYZABCD

F FGHIJKLMNOPQRSTUVWXYZABCDE

...

- Message: ATTACKATDAWN
- Pick a secret key DECAFDECAFDE
- Encrypted: DXV

1st letter in the message is shifted by 3, 2nd letter is shifted by 4, ...

	ABCDEFGHIJKLMNOPQRSTUVWXYZ
A	ABCDEFGHIJKLMNOPQRSTUVWXYZ
B	BCDEFGHIJKLMNOPQRSTUVWXYZA
C	CDEFGHIJKLMNOPQRSTUVWXYZAB
D	DEFGHIJKLMNOPQRSTUVWXYZABC
E	EFGHIJKLMNOPQRSTUVWXYZABCD
F	FGHIJKLMNOPQRSTUVWXYZABCDE

. . .

- Message: ATTACKATDAWN
- Pick a secret key DECAFDECAFDE
- Encrypted: DXVAHNEVDFZR

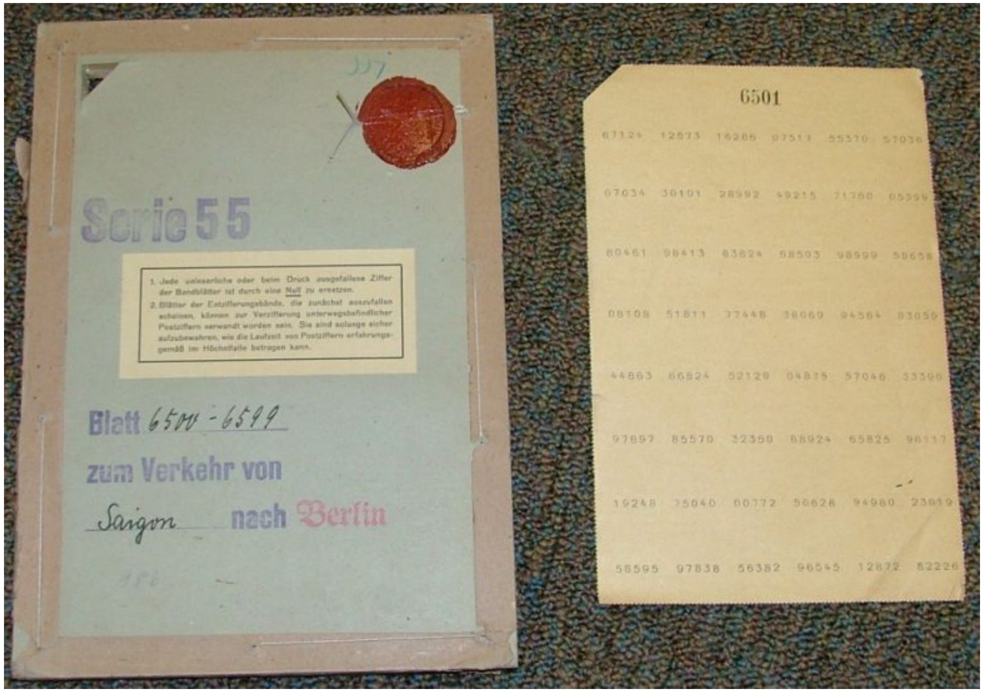
1st letter in the message is shifted by 3, 2nd letter is shifted by 4, …

Vernam Cipher

- Vigenère cipher was broken by Charles Babbage in the mid 1800s by exploiting the repeated key
  - The length of the key determines the cycle in which the cipher is repeated.
- Vernam cipher: make the key the same length as the message; Babbage's analysis doesn't work.

One-time Pads

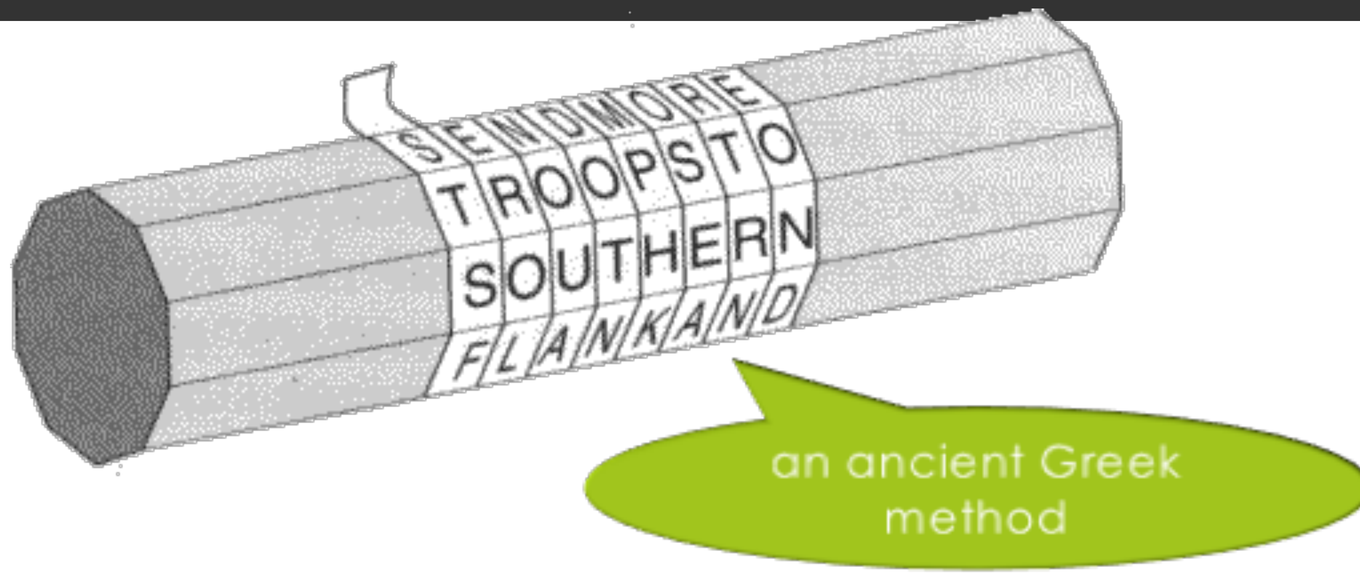
- Vernam cipher is commonly referred to as a one-time pad.



Alice and Bob have identical “pads” (shared keys)

- If random keys are used one-time pads are unbreakable in theory.

Transposition ciphers



STSF…EROL...NOUA...DOTN…MPHK…OSEA…RTRN…EOND…

image:http://crypto.interactive-maths.com/simple-transposition-ciphers.html

Encryption in computing

fast computation makes encryption usable by all of us

Encryption in computing

- One-time pads impractical on the net (why?)
- Basic assumption: the encryption/decryption *algorithm* is known; only the key is secret (why?)
- Very complicated encryptions can be computed fast:
  - typically, elaborate combinations of substitution and transposition

HTTPS

- Security protocol for the Web, the peoples' encryption
- Purpose:
  - confidentiality (prevent eavesdropping)
  - message integrity and authentication (prevent "man in the middle" attacks that could alter the messages being sent)
- Techniques:
  - asymmetric encryption ("public key" encryption) to exchange secret key
  - certificate authority to obtain public keys
  - symmetric encryption to exchange actual messages

Symmetric vs. asymmetric encryption

- **Symmetric** (shared-key) encryption: commonly used for long messages
  - Often a complicated mix of substitution and transposition encipherment
  - Reasonably fast to compute
  - Requires a shared secret key usually communicated using (slower) *asymmetric encryption*
- **Asymmetric** encryption: different keys are used to encrypt and to decrypt

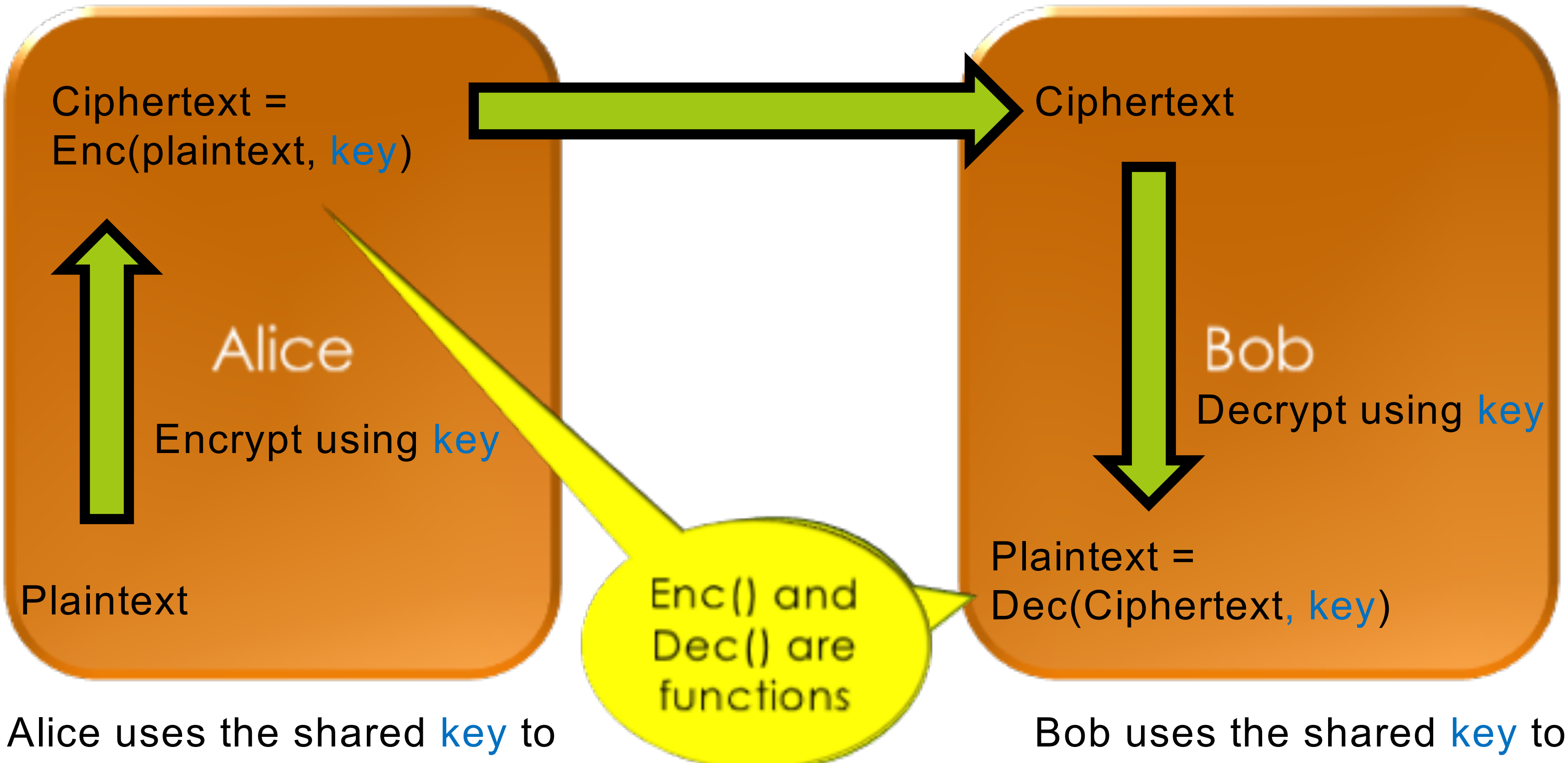
Keyspace

- *Keyspace* is jargon for the number of possible secret keys, for a particular encryption/decryption algorithm

- Number of bits per key determines *size of keyspace*
  - important because we want to make *brute force attacks* infeasible
  - brute force attack: run the (known) decryption algorithm repeatedly with **every possible key** until a sensible plaintext appears

- Typical key sizes: several hundred bits

Symmetric (Shared Key) Encryption



Alice uses the shared key to encrypt the plaintext to produce the ciphertext

Bob uses the shared key to decrypt the ciphertext to recover the plaintext

Establishing Shared Keys

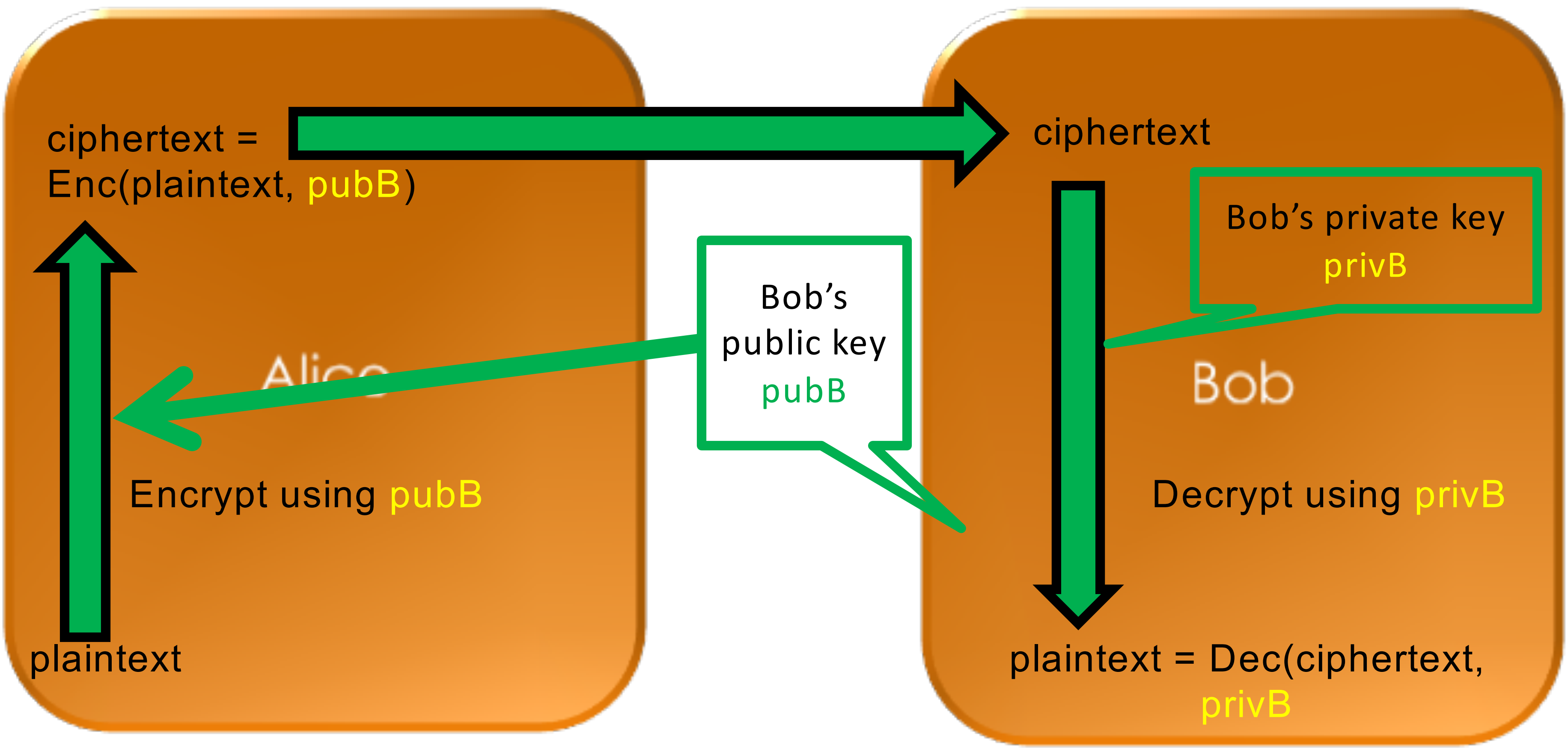
- Problem: how can Alice and Bob secretly agree on a key, using a public communication system?

- Solution: asymmetric encryption based on *number theory*
  - Alice has one secret, Bob has a different secret; working together they establish a shared secret
  - Examples: Diffie-Hellman key exchange, RSA public key encryption

One type of asymmetric encryption: RSA

- Common encryption technique for transmitting symmetric keys on the Internet (https, ssl/tls)
  - Named after its inventors: Rivest, Shamir and Adleman
  - Used in https (you know when you're using it because you see the URL in the address bar begins with http**s**://)

Asymmetric Public Key Encryption



Alice uses Bob's public key to encrypt the plaintext to produce the ciphertext

Bob uses his private key to decrypt the ciphertext to recover the plaintext

How RSA works

- First, we must be able to represent any message as a single number (it may already be a number as is usual for a symmetric key)
- For example:

A T **T** A **C** K **A** T **D** A **W** N

0120**20**01**03**11**01**20**04**01**23**14

Public and Private Keys

- Every receiver has a public key (e, n) and a private key (d, n).

used for encryption

used for decryption

- The transmitter encrypts a (numerical) message ciphertext C using the receiver's public key:

$$M^{e} \text{ modulo } n \rightarrow C \quad \textit{(ciphertext)}$$

- The receiver decodes the encrypted message C to get the original message M using the private key (which no one else knows).

$$C^{d} \text{ modulo } n \rightarrow M \quad \textit{(plaintext)}$$

RSA Example

- Alice's Public Key: (3, 33) (e = 3, n = 33)
- Alice's Private Key: (7, 33) (d = 7, n = 33)
  - Usually these are really huge numbers with many hundreds of digits!
- Bob wants to send the message 4
  - Bob encrypts the message using *e* and *n*:
    4^3 modulo 33 → 31 ... Bob sends 31
- Alice receives the encoded message 31
  - Alice decrypts the message using d and n:
    31^7 modulo 33 → 4

Generating *n*, *e* and *d*

- *p* and *q* are (big) random primes.
- $n = p \times q$
- $\varphi = (p - 1)(q - 1)$
- *e* is small and relatively prime to φ
- *d*, such that: $e \times d \bmod \varphi = 1$

$p = 3, q = 11$

$n = 3 \times 11 = 33$

$\varphi = 2 \times 10 = 20$

$e = 3$

$3 \times d \bmod 20 = 1$

$d = 7$

Usually the primes are huge numbers--hundreds of digits long.

Cracking RSA

- Everyone knows (e, n). Only Alice knows d.
- If we know e and n, can we figure out d?
  - If so, we can read secret messages to Alice.
- We **can** determine d from e and n.
  - Factor n into p and q.
    $n = p \times q$
    $\varphi = (p - 1)(q - 1)$
    $e \times d = 1 \pmod{\varphi}$
  - We know e (which is public), so we can solve for d.
- But **only** if we can factor n

RSA is safe (for now)

- Suppose someone can factor my 5-digit *n* in 1 ms,
- At this rate, to factor a 10-digit number would take 2 minutes.
- … to factor a 15-digit number would take 4 months.
- … 20-digit number … 30,000 years.
- … 25-digit number… 3 billion years.
- We're safe with RSA! (at least, from factoring with digital computers)

Certificate Authorities

- **How do we know we have the right public key for someone?**
- *Certificate Authorities* sign digital certificates indicating authenticity of a sender who they have checked out in the real world.
- Senders provide copies of their certificates along with their message or software.
- But can we trust the certificate authorities? (only some)

Encryption is not security!

It’s just a set of techniques

How (in)secure is the Internet?

- The NSA has a budget of $11B; we know from Edward Snowden how some of it is used

- Corporations and criminals also spy on us

- What can go wrong?
  - Insecure pseudo-random number generators
  - Untrustworthy certificate authorities
  - Malware
  - “Social engineering” attacks like phishing
  - Deliberately built-in insecurity in crypto products
  - Physical tapping of Internet routers

Security is an unsolved problem

Your cyber systems continue to function and serve you not due to the expertise of your security staff but solely due to the sufferance of your opponents.

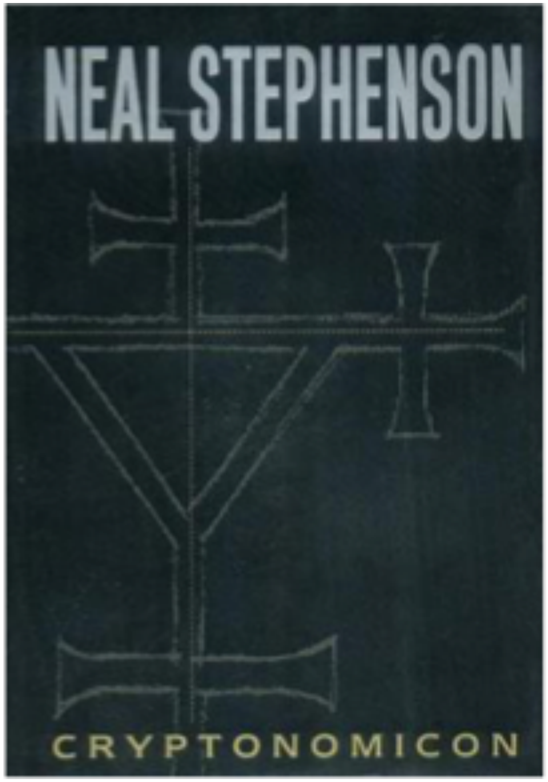
– former NSA Information Assurance Director Brian Snow (quoted by Bruce Schneier,
https://www.schneier.com/blog/archives/2013/03/phishing_has_go.html)

Summary

- Cryptography is cool mathematics and protocol design
- But cryptography is not security, only a set of techniques
- Security is a broader issue involving
  - Other technology
  - Social and legal factors

“Only amateurs attack machines; professionals target people” –Bruce Schneier

Two closing thoughts



Use Signal...