CSE 380
Computer Operating Systems

Instructor: Insup Lee and Dianna Xu

University of Pennsylvania
Fall 2003
Lecture Note: Security

Preface

- Early (unix systems) security
  - Security by obscurity
  - Those that know enough to break the system also know enough not to
- RTM
  - The Great Internet Worm of 1988
  - Devastating watershed event in hacker history
  - First awareness of internet security
- Legendary literatures:
  - *Hackers – Steven Levy*
  - *Cyberpunk – Hafner and Markoff*
  - *The Cuckoo's Egg – Clifford Stoll*
  - *The Jargon File*

Hackers vs Crackers

- The word hack doesn't have 69 different meanings
  - an appropriate application of ingenuity
  - a creative/brilliant practical joke
- Legendary hacks are revered as urban folklores
  - The element of cleverness
  - A flare for classic hacker's humor and style, which includes references to Adams, Tolkien as well as jargons
  - Mostly harmless
  - Caltech/MIT football pranks
  - Robin Hood/Friar Tuck against Xerox
- There is no cure against bored students

Robin Hood/Friar Tuck

!X id1

id1: Friar Tuck... I am under attack! Pray save me!
id1: Off (aborted)

id2: Fear not, friend Robin! I shall rout the Sheriff
of Nottingham's men!

id1: Thank you, my good fellow!

Terminology

- Vulnerability (weakness/defects that can be exploited)
  - Ill-chosen passwords
  - Software bugs
  - Communication without encryption
  - Incorrect set-ups
- Attack (ways of exploiting vulnerability)
  - Password crackers
  - Viruses and worms
  - Denial of service
- Intruders (adversaries that try to attack)
  - Terrorists
  - Espionage
  - Hackers

Security Goals

- Data Confidentiality
  - Keep data and communication secret
  - Privacy of personal financial/health records etc
  - Military and commercial relevance
- Data Integrity
  - Protect reliability of data against tampering
  - Can we be sure of the source and content of information?
- System Availability
  - Data/resources should be accessible when needed
  - Protection against denial of service attacks

Sample Tools

- Cryptography
  - Can ensure confidentiality and integrity
  - Typically used for authentication
- Firewalls, passwords, access control
  - Authorization mechanisms
- Operating systems
  - Resource allocation
  - Monitoring and logging for audits
- Java bytecode verifier
  - Memory safety against malicious/defective code

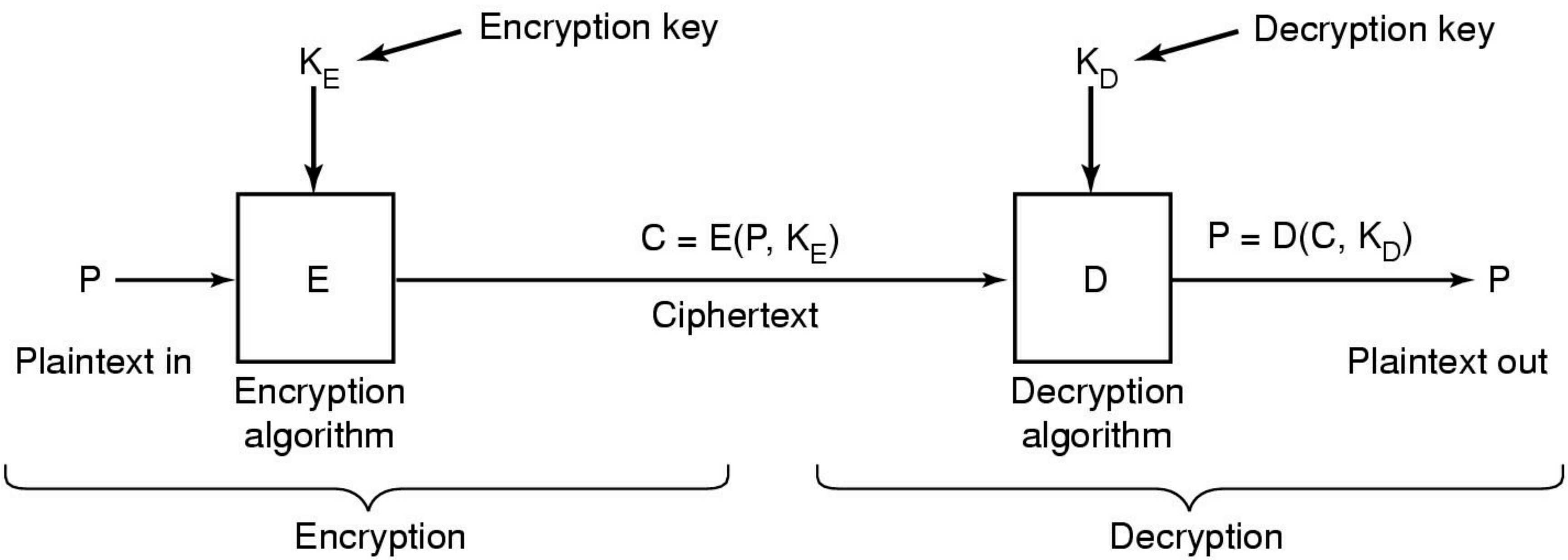
We do not have adequate technology today!

Basics

Terminology

- *Authentication:* Verifying identity of sender and/or message integrity
- *Integrity:* Message tampering detection
- *Plaintext:* Original message
- *Ciphertext:* Encrypted message
- *Key*: Input for en- and decryption algorithm
- *Encryption:* Plaintext + Key → Ciphertext
- *Decryption:* Ciphertext + Key → Plaintext

Basic Set-up of Cryptography



Relationship between the plaintext and the ciphertext

Encryption Algorithms

Symmetric

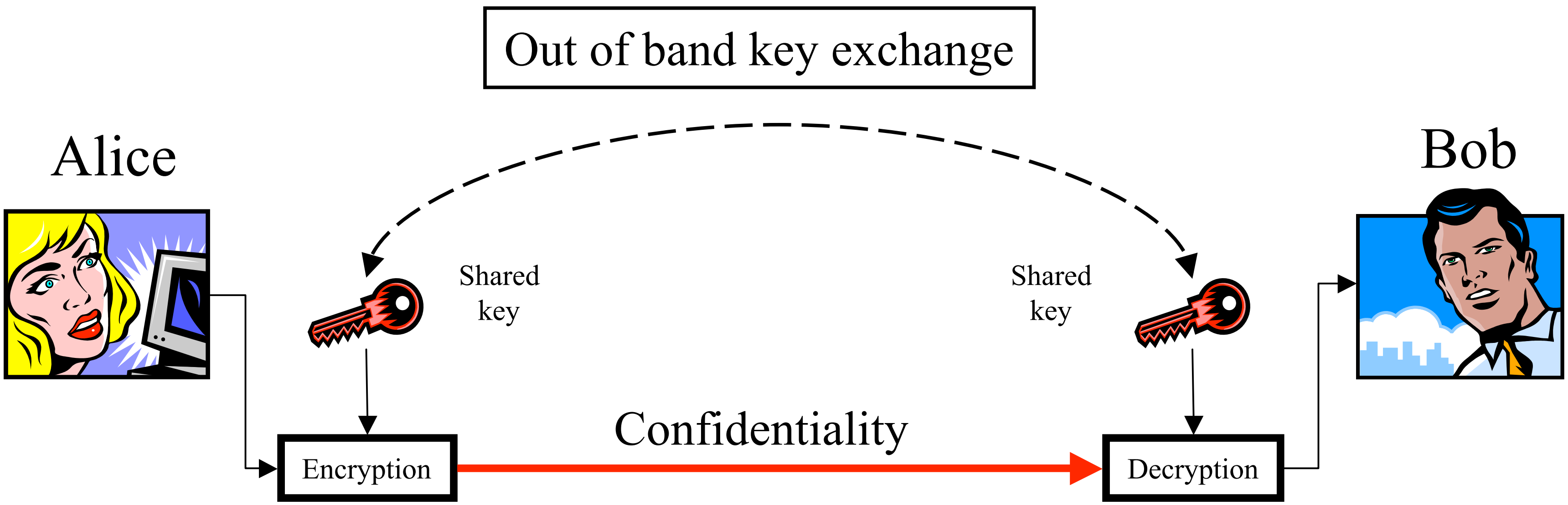
- Encryption and decryption use the same key
- Key must be secret (secret key)
- Best known: DES, AES, IDEA, Blowfish, RC5

Asymmetric

- Also known as Public Key Encryption
- Encryption and decryption keys *different*

DES – Data Encryption Standard, IDEA – International Data Encryption Algorithm, AES – Advanced Encryption System

Symmetric Encryption



Monoalphabetic Ciphers

- Classical way of encoding text strings (Caesar Cipher)
- Permutation of the alphabet (rot13)
- The key for decoding is the inverse permutation
- Encoding and decoding are efficient
- Theoretically sound: the number of permutations of ASCII alphabet is VERY large (128!), and an intruder cannot possibly try out all possible permutations to decipher
- Main problem: Any human language has distinct frequent letter (e.g. vowels) combos
  - E.g. e is the most common letter in English text, th is the most common sequence of adjacent symbols
  - Given enough cipher text, one doesn't need to be Shelock Holmes to break the code

Secret-Key Cryptography

- Sender and receiver share the secret key
- This is also called symmetric key cryptography
- A popular scheme for many years: DES (Data Encryption Standard) promoted by NSA
  - Key is 56 bits (extended to 64 bits using 8 parity bits)
  - Input data is processed in chunks of 64-bit blocks, by subjecting to a series of transformations using the key
- Distribution of keys is a problem

Asymmetric Encryption

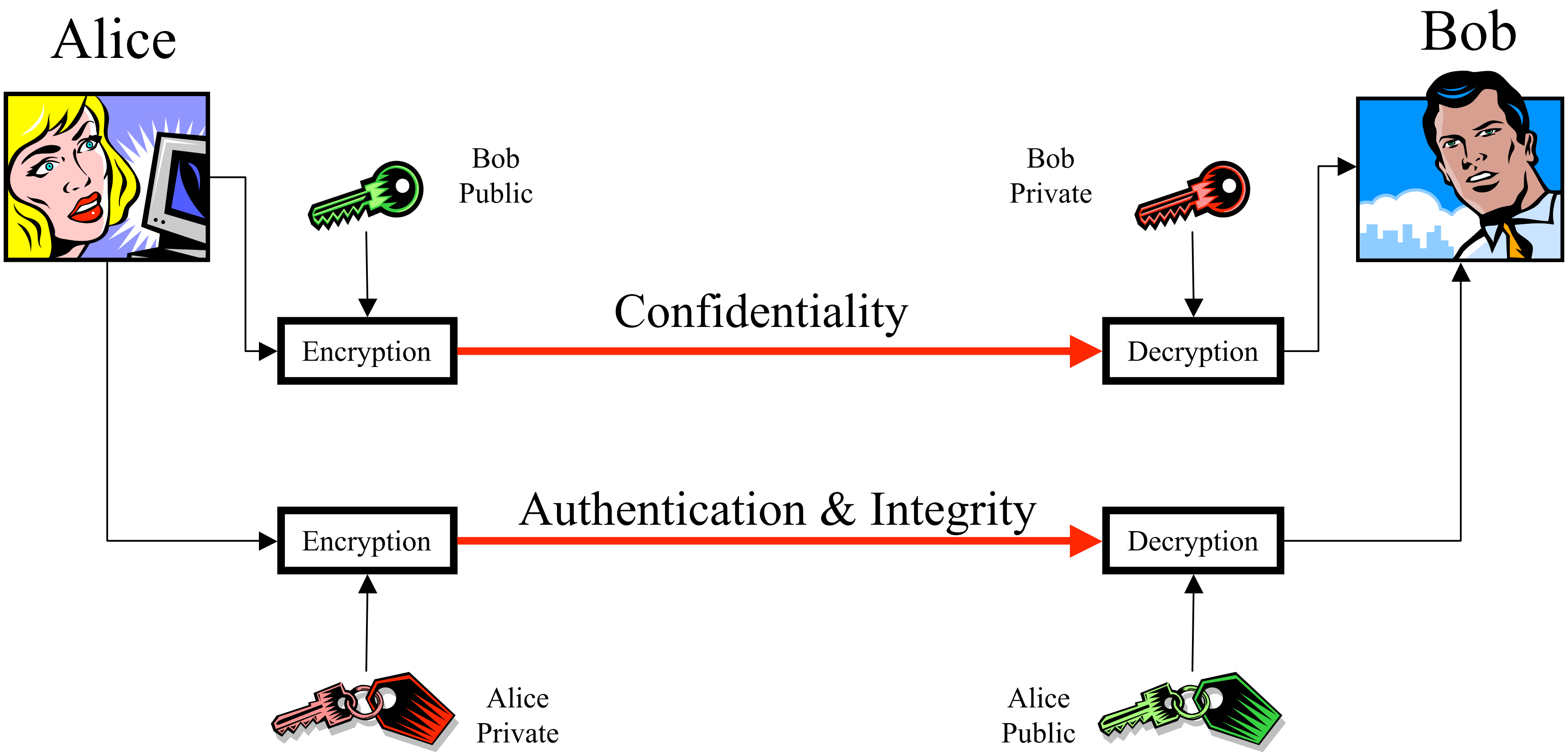
- Two complementary keys
  - Private key (kept secret)
  - Public key (published)
- Private key VERY difficult to compute from public key
- Encryption with one key can only be reversed with the other key
- Used in PGP (Pretty Good Privacy) & PKI (Public Key Infrastructure)
- Best known RSA & ECC, DSA for signatures

RSA Rivest Shami Adleman, ECC – Eliptic Curve Cryptography, DSA – Digital Signature Algorithm

One-Way Functions

- Function such that given formula for f(x)
  - easy to evaluate y = f(x) given x
- But given y
  - computationally infeasible to find x
- There is a rich theory of one-way functions
  - Many candidates proposed
  - None of them “proved” to be one way
  - Existence of one-way functions linked to encryption, random number generators, (and other crypto concepts) in a precise sense

Asymmetric Encryption cont'd



Public-Key Cryptography

- All users pick a public key/private key pair
  - publish the public key
  - private key not published
- Public key is the encryption key
  - To send a message to user Alice, encrypt the message with Alice's public key
- Private key is the decryption key
  - Alice decrypts the ciphertext with its private key
- Popular schemes (1970s): Diffie-Hellman, RSA

More on RSA

- Introduced by Rivest, Shamir, and Adleman in 1979
- Foundations in number theory and computational difficulty of factoring
- Not mathematically proven to be unbreakable, but has withstood attacks and analysis
  - Ideally, we would like to prove a theorem saying "if intruder does not know the key, then it cannot construct plaintext from the ciphertext by executing a polynomial-time algorithm"
- Public and private keys are derived from secretly chosen large prime numbers (512 bits)
- Plaintext is viewed as a large binary number and encryption is exponentiation in modulo arithmetic
- Intruder will have to factor large numbers (and there are no known polynomial-time algorithms for this)
  - 2002's major result: polynomial-time test to check if a number is prime

Hash Functions

- Produce *hash values* for data access or security
- *Hash value*: Number generated from a string of text
- Hash is substantially smaller than the text itself
- Unlikely that other text produces the same hash value (collision resistance)
- Unidirectional (cannot calculate text from hash)
- Provides: Integrity & Authentication
- Best known: SHA-1 & MD5

SHA – Secure Hash Algorithm, MD5 – Message Digest

Digital Signatures

- How can Alice *sign* a digital document ?
- Let S(A,M) be the message M tagged with Alice's signature
- No forgery possible: If Alice signs M then nobody else can generate S(A,M)
- Authenticity check: If you get the message S(A,M) you should be able to verify that this is really created by Alice
- No alteration: Once Alice sends S(A,M), nobody (including Alice) can tamper this message
- No reuse: Alice cannot duplicate S(A,M) at a later time

Digital Signatures with Public Keys

- Suppose K is public key and k is private key for Alice, and encryption/decryption is commutative:

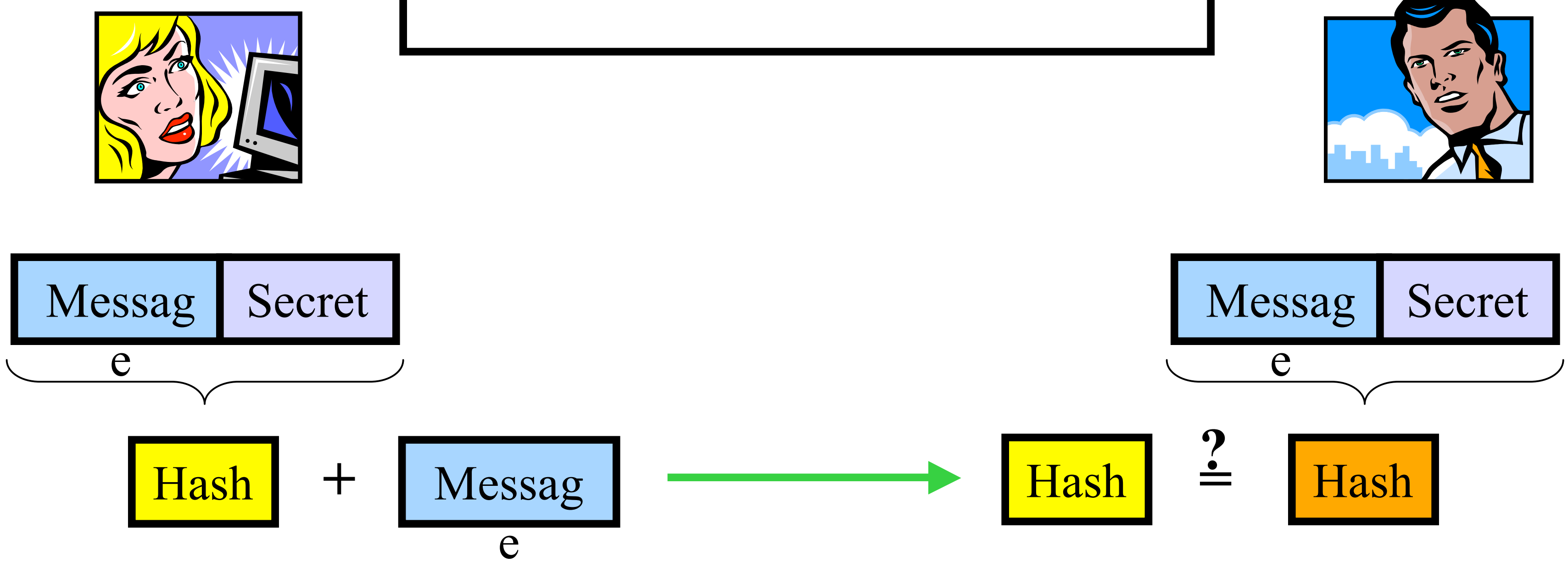
  D(E(M,K), k) = E(D(M,k),K)=M
- To sign a message M, Alice simply sends D(M,k)
- Receiver uses Alice's public key to compute E(D(M,k),K), to retrieve M
  - Authenticity of signature because only Alice knows the private key k
- RSA encryption does satisfy the required commutativity
- To ensure "no reuse" and "no alteration" the message must include a timestamp
- The scheme is made more efficient by computing D(H(M),k), where H(M) is the *secure hash* of M
  - Hashing gives a constant size output
  - Hard to invert

Hash Functions cont'd

Provides signatures with *shared secret*

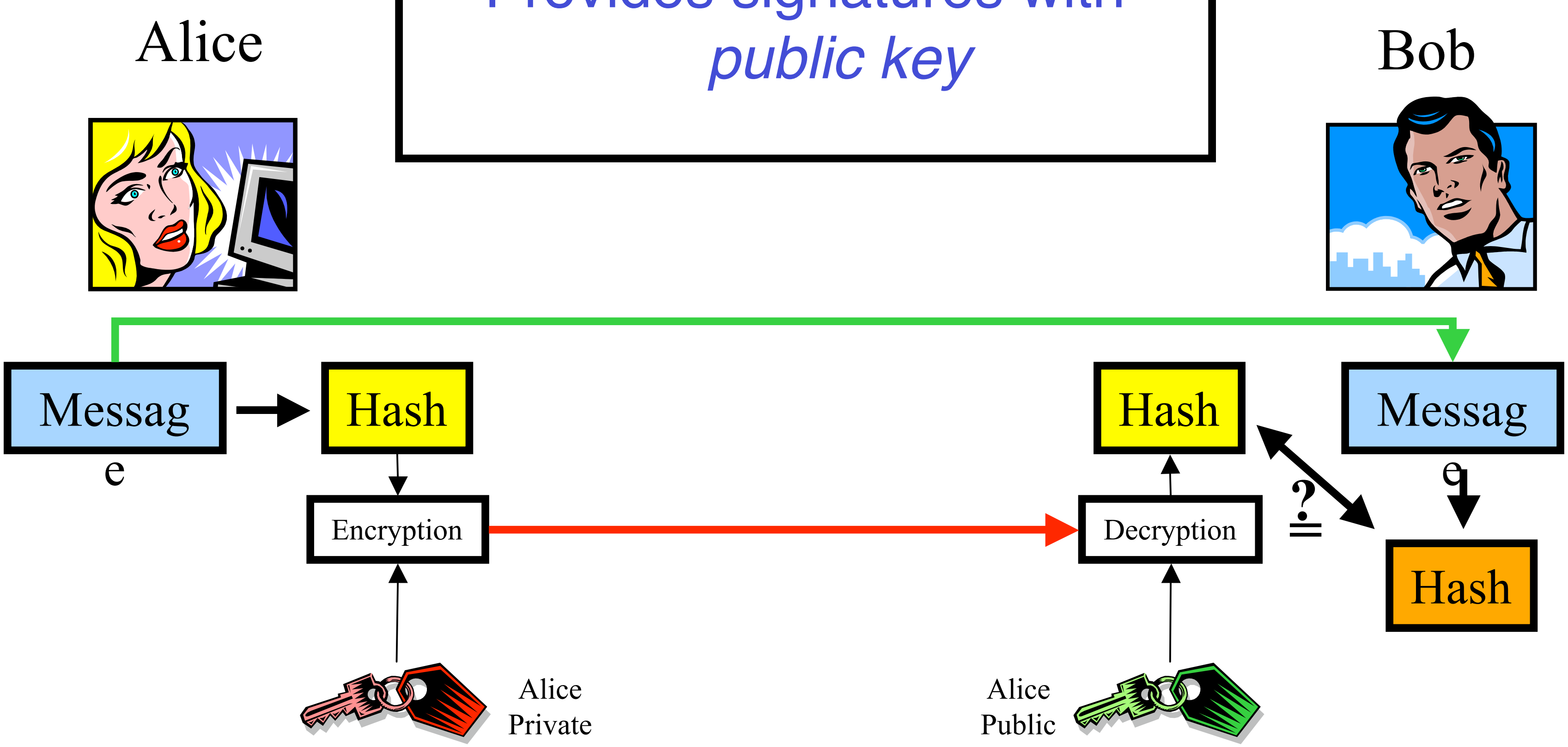
Alice

Bob



Hash Functions cont'd

Provides signatures with *public key*



PKI in a Nutshell

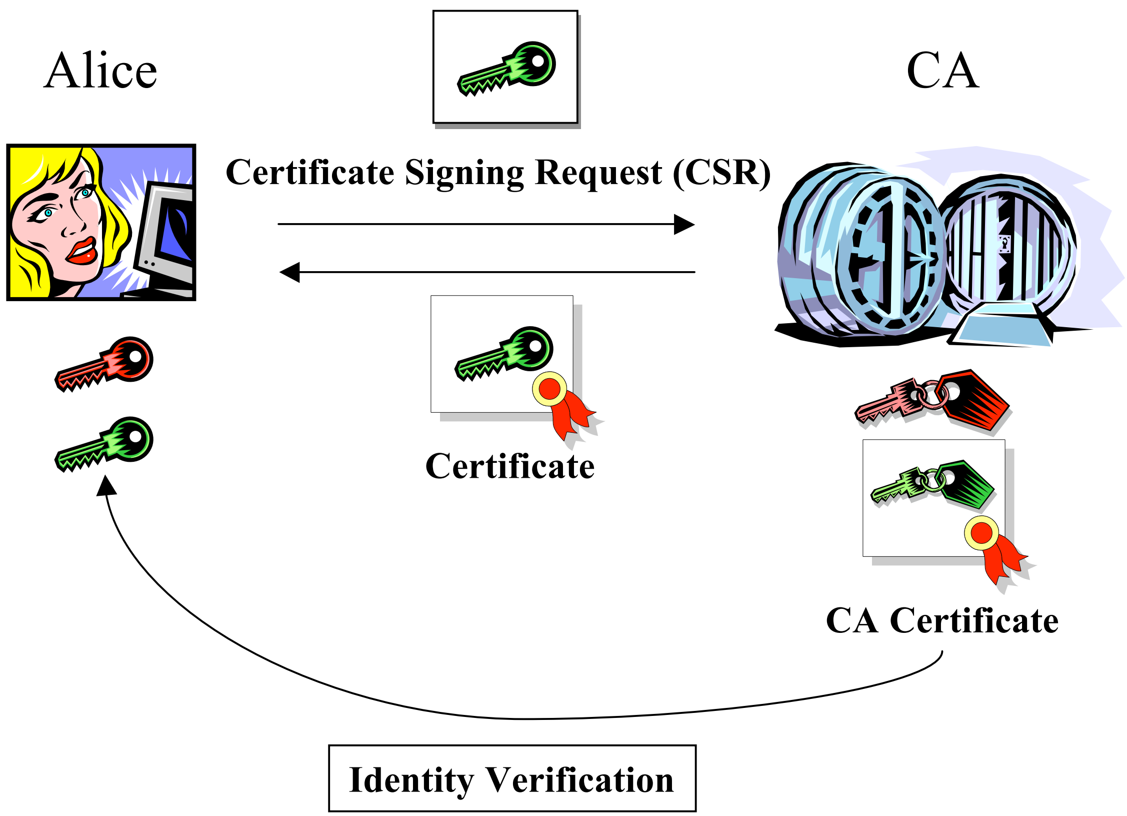
PKI (*P*ublic *K*ey *I*nfrastructure) based on

- Certificates (X.509)
- Chain of trust (usually hierarchy)

Certificates

- Public keys signed by a trusted 3rd party
  CA = *C*ertificate *A*uthority
- Certificate is public as well
- Different types for people, web server, …

Certificate creation



User Authentication

Authentication is the process of determining which user is making a request

Basic Principles. Authentication must identify:

1. Something the user knows (e.g. password)
2. Something the user has (e.g. ID card)
3. Something user is (e.g. retina scan)

Humans are the weakest link

Passwords

- The most commonly used way of authentication
- Vulnerabilities
  - Stealing passwords
  - Poorly chosen passwords that are easy to guess
  - Attacks that search through password directories
- If you were to guess passwords, how would you go about doing that?
- Survey of passwords by Morris&Thomson: could guess 86% of all passwords
  - 15 single ASCII letters
  - 72 two ASCII letters
  - 464 three ASCII letters
  - Words from dictionary, names of people/streets ….

Systems are easy to crack!

LBL> telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD: root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL

- How a cracker broke into LBL
  - a U.S. Dept. of Energy research lab

Password Attacks

- Deadly combo:
  - War dialers / password guessing
- Once entrance to a system is gained:
  - password file
  - packet sniffer
  - rsh/rlogin into other machines with known usr/passwd combo
- Social Engineering

Unix: /etc/passwd

- Passwords stored in a file system are vulnerable to automated attacks
  - At first Unix was implemented with a password file holding the actual passwords of users, but with only root permissions.
- This had many vulnerabilities
  - Copies were made by privileged users
  - Copies were made by bugs: classic example posted password file on daily message file

Improvements to First Approach

- Enforce password rules
  - Makes the passwords harder to guess or crack with dictionaries
  - Problems?
- Hashing and encryption: use password to create a key, then hash based on the DES algorithm for encryption
  - Speed OK for legitimate users
  - Takes longer to do automatic search
- Password files contains these encrypted entries
- Intruder cannot figure out the passwords just by gaining access to password file, but can keep guessing passwords, apply hash/encryption and compare the results to entries in password file

Add Salt

- "Salt" the passwords by adding random bits.
  - Makes dictionary attacks more expensive.
  - Decreases the likelihood that two identical passwords will appear as identical entries in the password file.
- 12 bit salt results in 4,096 versions of each password.
- /etc/passwd entry:

user_id	Salt	Hash(salt + passwd)	…

- How does this help?

Hash-based 1-time Passwords

- Goal: Can the password be different in every session?
  - code books
- Scheme used for remote logins based on one-way hash functions
- One-time setup.
  - User chooses a password w
  - Fixes a constant t for the number of times the authentication can be done using password w
  - User declares the password $H^t(w)$ to the system the first time

$$H(H(H\ldots(H(w))\ldots))$$

One time passwords

- Initially, the computer stores, with user's login-id, password $p=H^t(w)$ and session number s=0
- After i sessions the computer has $p=H^{t-i}(w)$ and s=i
- At the time of login, computer sends i to the user
- User computes new password $q=H^{t-i-1}(w)$ and sends it to the computer
- The computer checks that H(q)=p, and if so, allows the login (and updates local entries to q and i+1)
- Important property: given q, it is easy to compute H(q), but if intruder had stolen p in the last session, it cannot produce q
  - H is a one-way hash function, hard to invert

Operating System Security

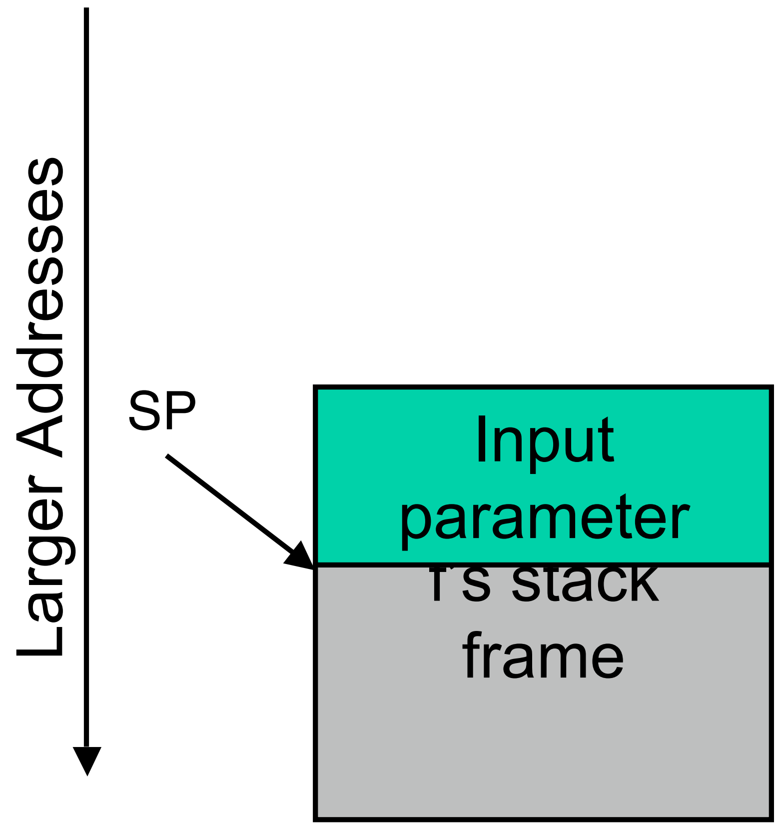
- Trojan horses
  - Free programs available to be downloaded and executed
  - Common trick: place altered versions of utility programs in user directories
- Login Spoofing
  - Simulate the login session to acquire passwords
- Logic Bomb
- Trap Doors
  - System programmer writes code to bypass normal checks
  - Insider knowledge to exploit these intentional vulnerabilities

Buffer Overflow Attacks

- > 50% of security incidents reported at CERT (see cert.org) are due to buffer overflow attacks
- C and C++ programming languages don't do array bounds checks
  - In particular, widely used library functions such as strcpy, gets
- Exploited in many famous attacks (read your Windows Service Pack notes)

C's Control Stack

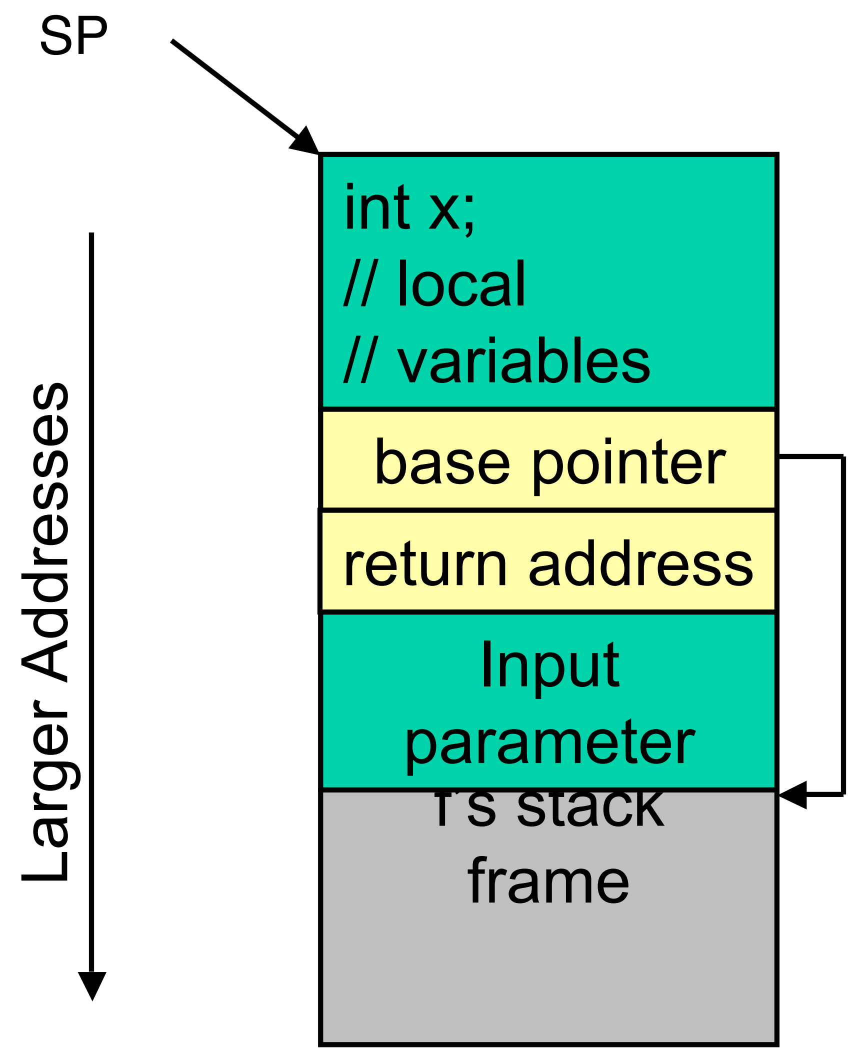
```
f() {
  g(parameter);
}

g(char *args) {
  int x;
  // more local
  // variables
  ...
}
```



Before calling g

C's Control Stack

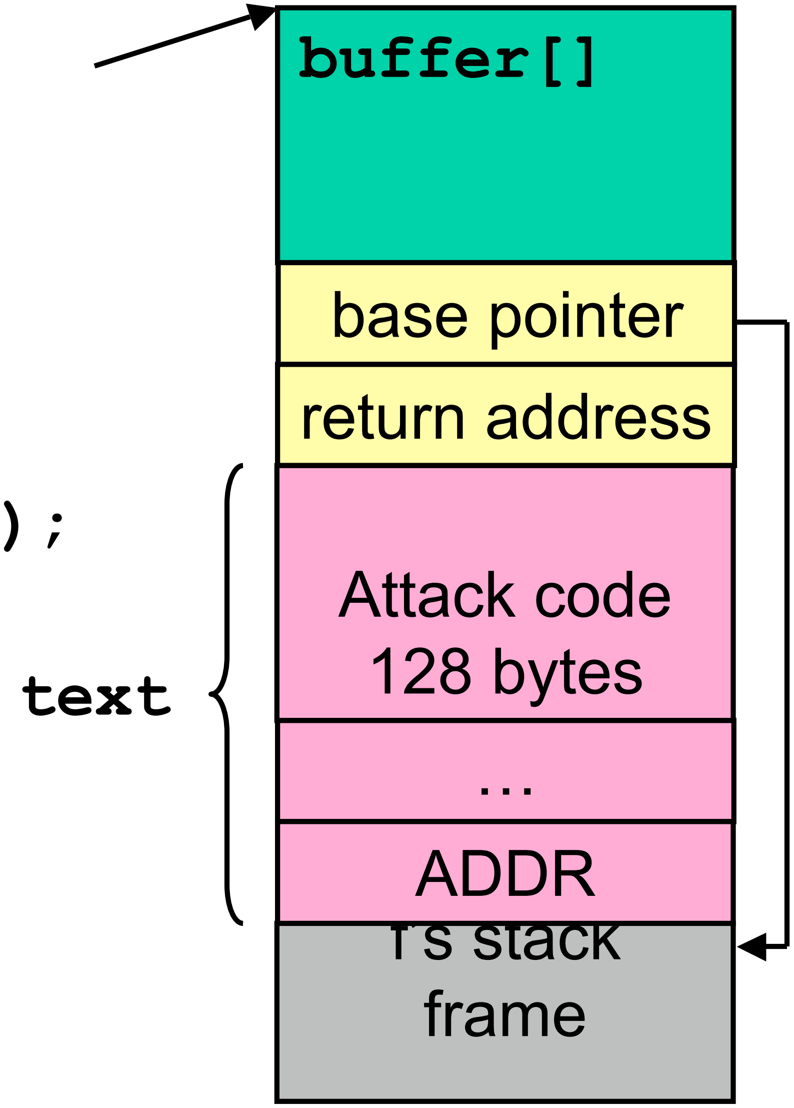
```
f() {
  g(parameter);
}

g(char *args) {
  int x;
  // more local
  // variables
  ...
}
```



After calling g

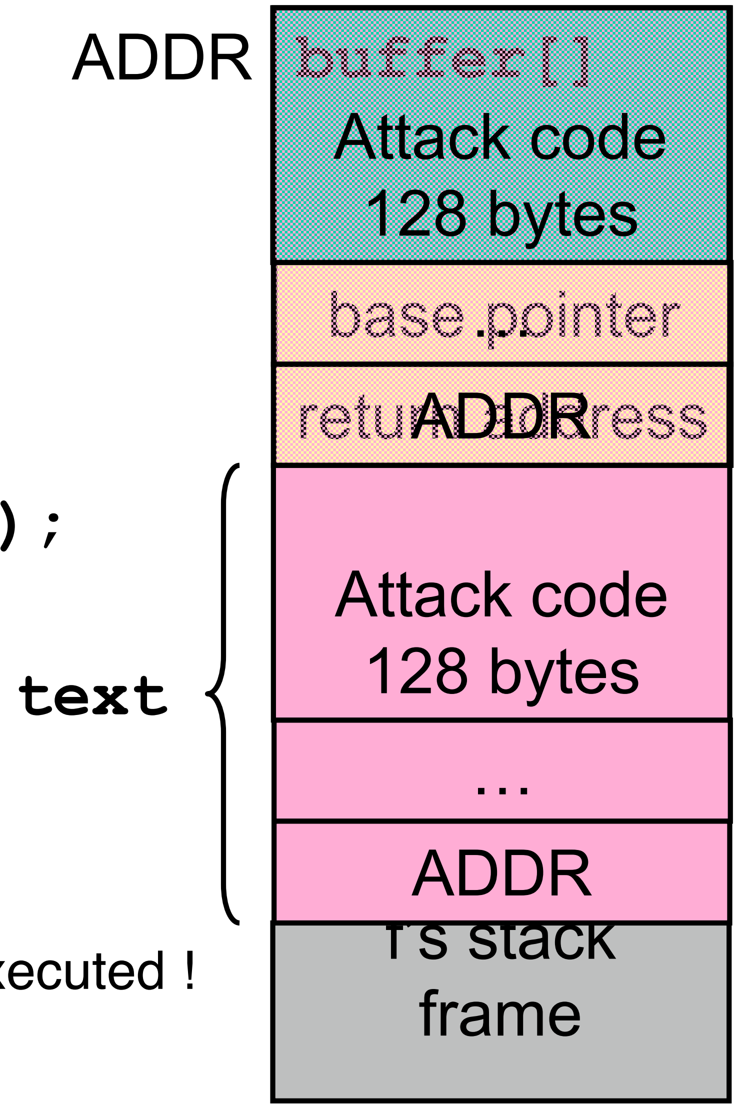
Buffer Overflow Example

```
g(char *text) {
  char buffer[128];
  strcpy(buffer, text);
}
```



Buffer Overflow Example

```
g(char *text) {
   char buffer[128];
   strcpy(buffer, text);
}
```



Upon return from g, attack code gets executed !

Solutions

- Don't write code in C
  - Use a safe language instead (Java, C#, …)
  - Not always possible (low level programming)
  - Doesn't solve legacy code problem
- Link C code against safe version of libc
  - May degrade performance unacceptably
- Software fault isolation
  - Instrument executable code to insert checks
- Program analysis techniques
  - Examine program to see whether "tainted" data is used as argument to strcpy

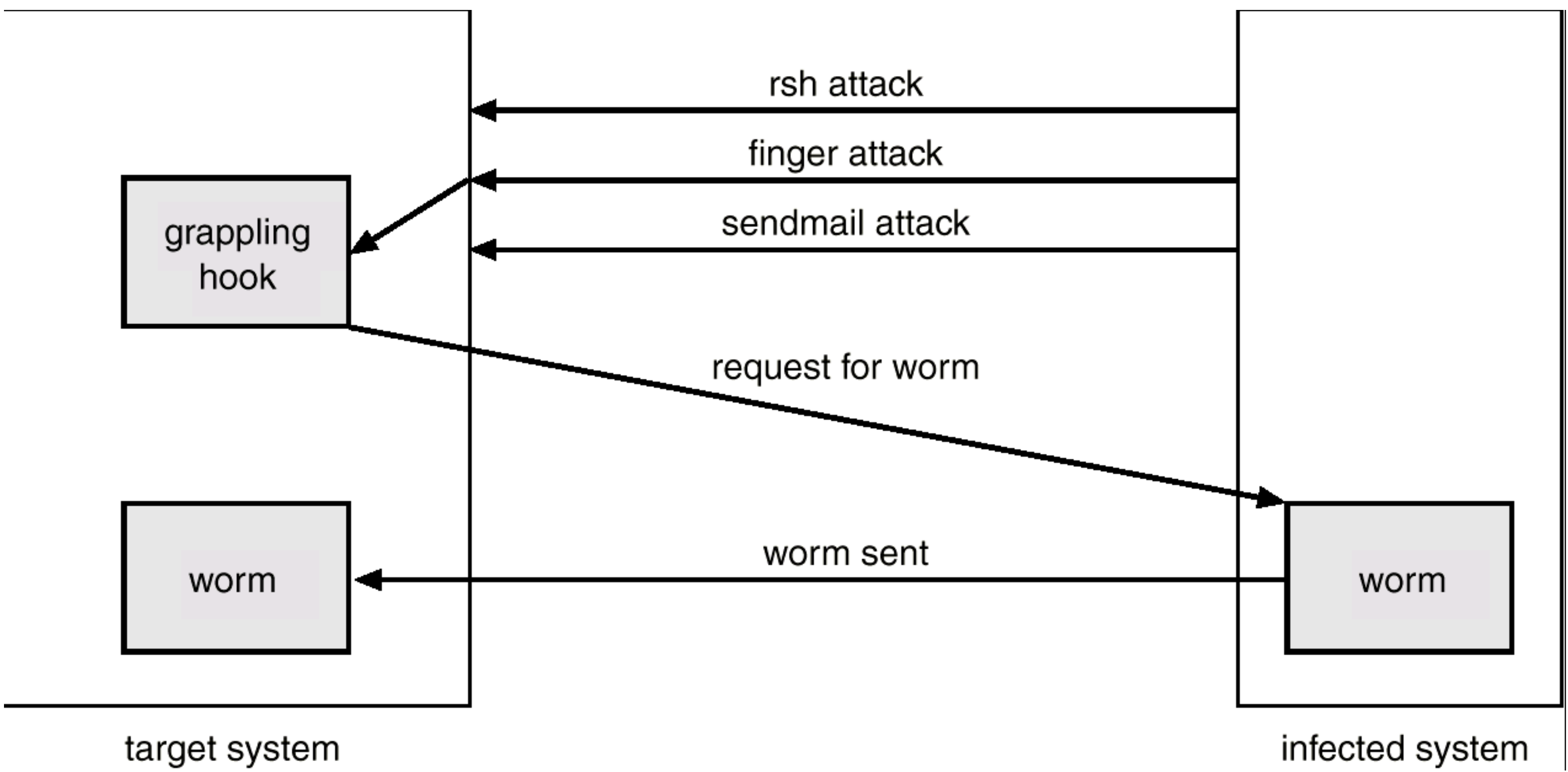
Avoiding Titanics

- Unix
  - lpr
  - link core to /etc/passwd
- Microsoft
  - code red (buffer overflow in IIS Indexing Service)
- Weathering actual attacks is the best way to make an OS safe
  - tiger teams
- System design should be public
- Keep the design simple

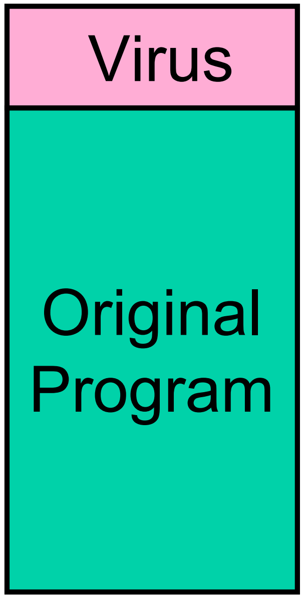
Network Security

- External threat
  - code transmitted to target machine
  - code executed there, doing damage
- Goals of virus writer
  - quickly spreading virus
  - difficult to detect
  - hard to get rid of
- Virus = program can reproduce itself
  - by attaching its code to another program
  - additionally, do harm
- Worm
  - self-replicating

The Morris Internet Worm



Virus Attachment: Append

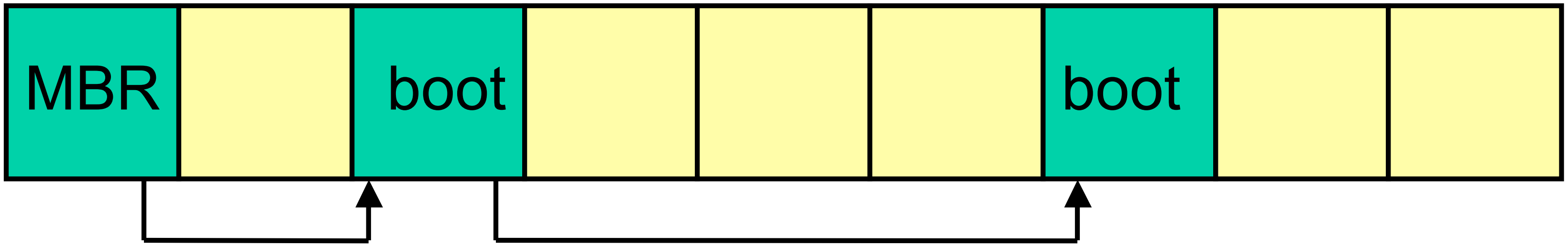


- Simplest case: insert copy at the end of an executable file
- Runs before other code of the program (by changing start address in header)
- Most common program virus

Kinds of Viruses

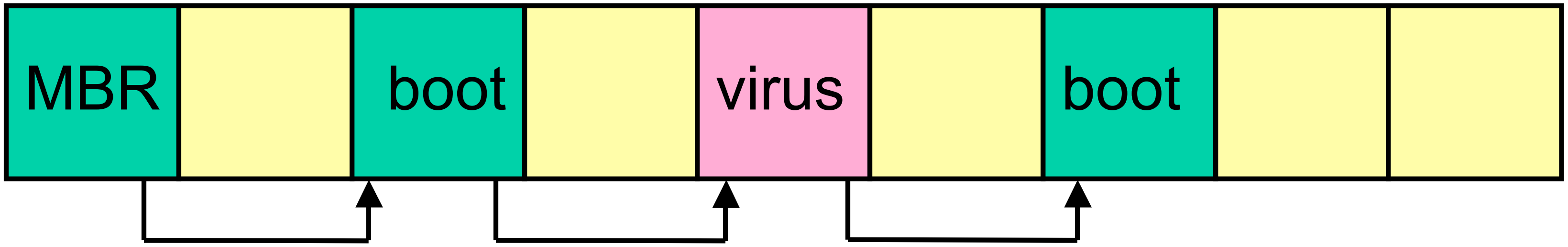
- Overwriting Viruses
  - Companion Viruses
  - Executable Viruses
- Parasitic Viruses
  - Cavity Viruses
- Memory-resident Viruses
  - System-call-trap Viruses
  - Software Viruses (Windows manager, explorer, etc)
- Boot Sector Viruses
- Device Driver Viruses
- Macro Viruses

Bootstrap Viruses



- Bootstrap Process:
  - Firmware (ROM) copies MBR (master boot record) to memory, jumps to that program
- MBR (or Boot Sector)
  - Fixed position on disk
  - "Chained" boot sectors permit longer Bootstrap Loaders

Bootstrap Viruses



- Virus breaks the chain
- Inserts virus code
- Reconnects chain afterwards

Why the Boot Sector?

- Automatically executed *before* OS is running
  - Also before detection tools are running
- OS hides boot sector information from users
  - Hard to discover that the virus is there
  - Harder to fix

- Any good virus scanning software scans the boot sectors

Macro Viruses

- Macros are just programs
- Word processors & Spreadsheets
  - Startup macro
  - Macros turned on by default
- Visual Basic Script (VBScript)

Melissa Virus

- Transmission Rate
  - The first confirmed reports of Melissa were received on Friday, March 26, 1999.
  - By Monday, March 29, it had reached more than 100,000 computers.
  - One site got 32,000 infected messages in 45 minutes.
- Damage
  - Denial of service: mail systems off-line.
  - Could have been much worse

Melissa Macro Virus

- Implementation
  - VBA (Visual Basic for Applications) code associated with the "document.open" method of Word
- Strategy
  - Email message containing an infected Word document as an attachment
  - Opening Word document triggers virus if macros are enabled
- Propagation
  - Sends email message to first 50 entries in every Outlook address book readable by the user executing the macro

“I Love You” Virus/Worm

- Infection Rate
  - At 5:00 pm EDT May 8, 2000, CERT had received reports from more than 650 sites
  - > 500,000 individual systems
- VBScript
- Propagation
  - Email, Windows file sharing, IRC, USENET news
- Signature
  - An attachment named
    "LOVE-LETTER-FOR-YOU.TXT.VBS"
  - A subject of "ILOVEYOU"
  - Message body: "kindly check the attached LOVELETTER coming from me."

Love Bug Behavior

- Replaced certain files with copies of itself
  - Based on file extension (e.g. .vbs, .js, .hta, etc)
- Changed Internet Explorer start page
  - Pointed the browser to infected web pages
- Mailed copies of itself
- Changed registry keys

Antivirus and Anti-Antivirus Techniques

- Scanning the disk for certain executables
  - hard to deal with polymorphic viruses
- Integrity checkers using checksums
- Behavioral checkers
- Virus avoidance
  - good OS
  - install only shrink-wrapped software
  - do not click on attachments to email
  - use antivirus software
  - frequent backups
- Recovery from virus attack
  - halt computer, reboot from safe disk, run antivirus