
CSC 369

Week 12/13: Security
Reading: Text, Chapter 9.1-9.5

This week

- Types of security threats
 - Protection basics
 - Intrusion
 - Malicious software
-
- Some principles of computer security

Security

- Computer Security
 - Techniques for **computing** in the presence of adversaries
 - Four requirements of security
 - **Confidentiality**: preventing unauthorized release of info
 - **Integrity**: preventing unauthorized modification of info
 - **Availability**: ensuring access to legitimate users
 - **Authenticity**: verifying the identify of a user
 - Protection is about providing all on a single machine
 - Usually considered the responsibility of the OS
- Cryptography
 - Techniques for **communicating** in the presence of adversaries

Types of Threats

- Security requirements are intended to thwart four corresponding types of threats
- Interception, or eavesdropping - attacker gains knowledge they should not have access to
 - **Loss of confidentiality**
 - Reading or copying files that attacker should not have access to
 - Intercepting network packets
 - IP packets include destination field in header. Well-behaved network cards ignore packets that are not intended for them, but network cards can be placed in "promiscuous mode" where all packets are accepted

More threats...

- Modification - attacker alters existing files, programs, network packets, etc.
 - Attack on integrity
 - Destruction of data is a special case
 - Attack on availability
- Theft of service - attacker installs daemon
 - Attack on availability
- Fabrication - attacker creates counterfeit objects (files, messages, etc.) which appear to come from a trusted source
 - Attack on authenticity

Vulnerabilities in the System

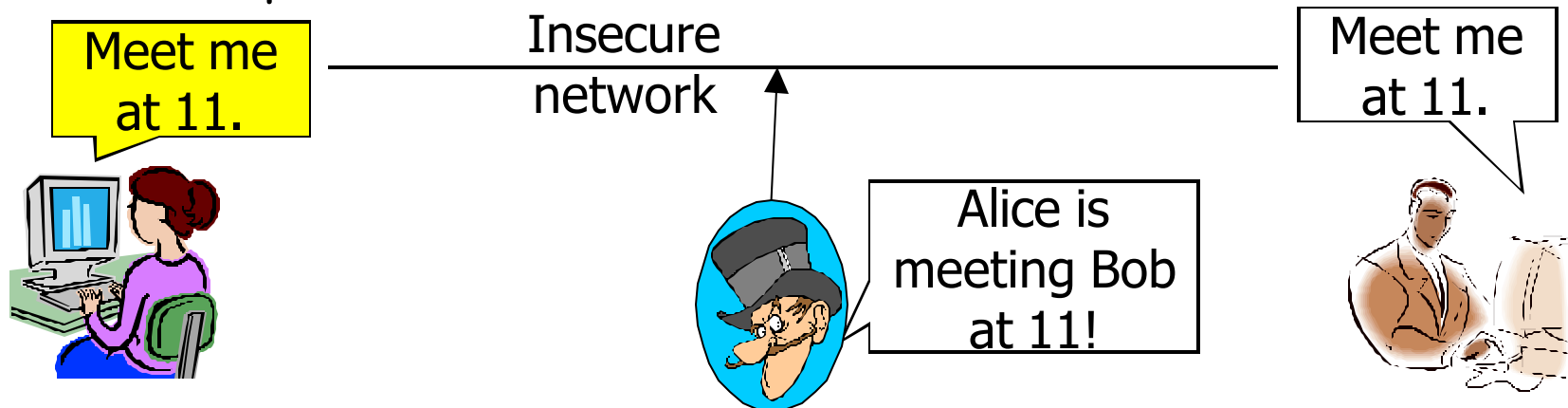
- Physical access
 - Unauthorized physical access makes it a lot easier to gain unauthorized digital access
- Humans
 - Who should you trust? How much?
- Operating system
 - Flaws in the system allows security protocols to be circumvented
- Networks
 - Data travels over unsecured communication lines, across multiple administrative domains
 - Focus on this level today

Network Communications

- Protecting “assets” on a single computer system with multiple users is hard, but dealing with networks is even harder
 - Multiple administrative domains
 - Users of network may not have common interests
- Two main categories of network attacks
 - Passive - eavesdropping or monitoring communications
 - Active - modification or tampering with the communication stream

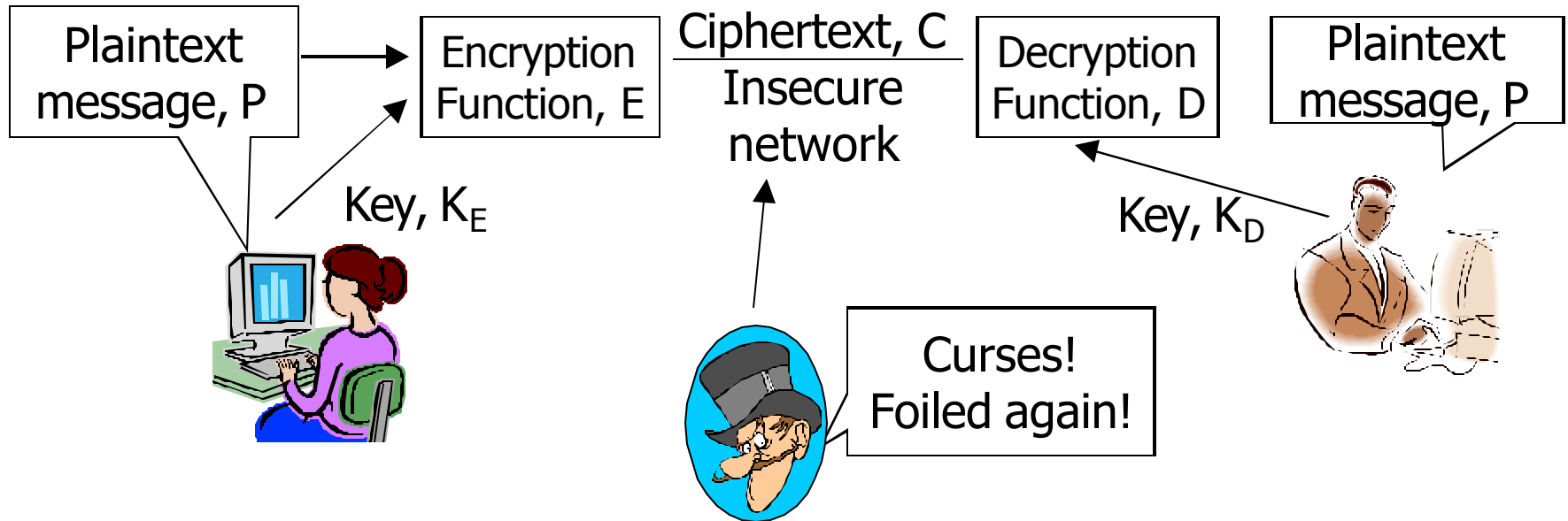
Passive Network Attacks

- **Eavesdropping** - read network packets that were not intended for attacker
 - Does not interfere with delivery of message to intended recipient



- Defense: obfuscate message contents so attacker gains no information from intercepting the message (i.e., encrypt the message)

Encryption basics



- $C = E(P, K_E)$
- $P = D(C, K_D) = D(E(P, K_E), K_D)$

Encryption Keys

- Keys used for encryption and decryption can be the same (symmetric, or secret-key) or different (public-key)
- Secret-key cryptography
 - ✓ Fast encryption/decryption algorithms are known
 - ✗ Distributing the shared secret key is a problem
- Public-key cryptography
 - Pair of keys, one to encrypt, one to decrypt
 - ✓ Encryption key can be published, decryption key is kept secret; knowledge of one key does not reveal other
 - ✗ Encryption/decryption algorithms ~1000X slower

More Passive Attacks

- **Traffic Analysis** - infer information based on sender/recipient of messages, size of messages, frequency of communication, etc.
 - Encryption doesn't help here
 - Defense: obfuscate communication pattern (i.e., make all messages the same length, all communications have same number of messages, etc.)

Active Network Attacks

- **Replay** - capture a legitimate message and re-send it later to produce unauthorized effect
 - e.g. unauthorized file accesses
 - Defence: "nonce" - messages include an item so they can't be reused
- **Modification of messages** - alter contents of message to change effect
 - Include message digest to detect tampering
- **Masquerade** - attacker pretends to be another entity or user
 - Often combined with another attack like replay

Message Digests

- Given some input data of arbitrary length, compute a fixed-length output
 - Relies on one-way function
 - Given input x , computing digest $y = f(x)$ is "easy"
 - Given y , finding x is computationally infeasible
 - So attacker cannot construct a new input that will have the same digest as the original
- To detect tampering, compare digest computed on received message against digest of sent message
 - Need to know original digest (same problem?)
- Sender can encrypt or sign digest only
 - Or distribute digest separately from message

Digital Signatures

- Provide a way to verify the origin and integrity of a message (i.e. authenticity)
- Sender includes something with the message (or modifies the message in some way) that could only be generated by the sender
- Public-key cryptography helps here
 - Confidentiality - encrypt message with recipients public key; only intended recipient can decrypt using their private key
 - Signatures - encrypt message with sender's private key; anyone can decrypt using sender's public key, thereby verifying that sender created the message
 - Typically, only a message digest is signed to reduce overhead

One more active attack

- Denial of service - system asset becomes unavailable to legitimate users
 - **Attack on availability**
 - Common network attacks - overload server with bogus requests so there are no resources for legitimate ones (e.g. TCP SYN flooding)
 - Extra evil - distributed DOS - compromise a set of computers and use them as "zombies" in a coordinated attack on the victim
 - How do you distinguish an attack from heavy load?
- Active attacks are harder to defend against
 - Need measures to detect and recover from them

Intrusion Detection

- Generally want to know when a system has been compromised
- Two main approaches
 - Signature-based detection - examine system activity or network traffic for patterns that match known attacks
 - Anomaly detection - identify patterns of "normal" behavior over time and detect deviations from those patterns
- Systems need records of user activity to help detect and recover from intrusions
 - These records have to be protected from tampering by the intruder

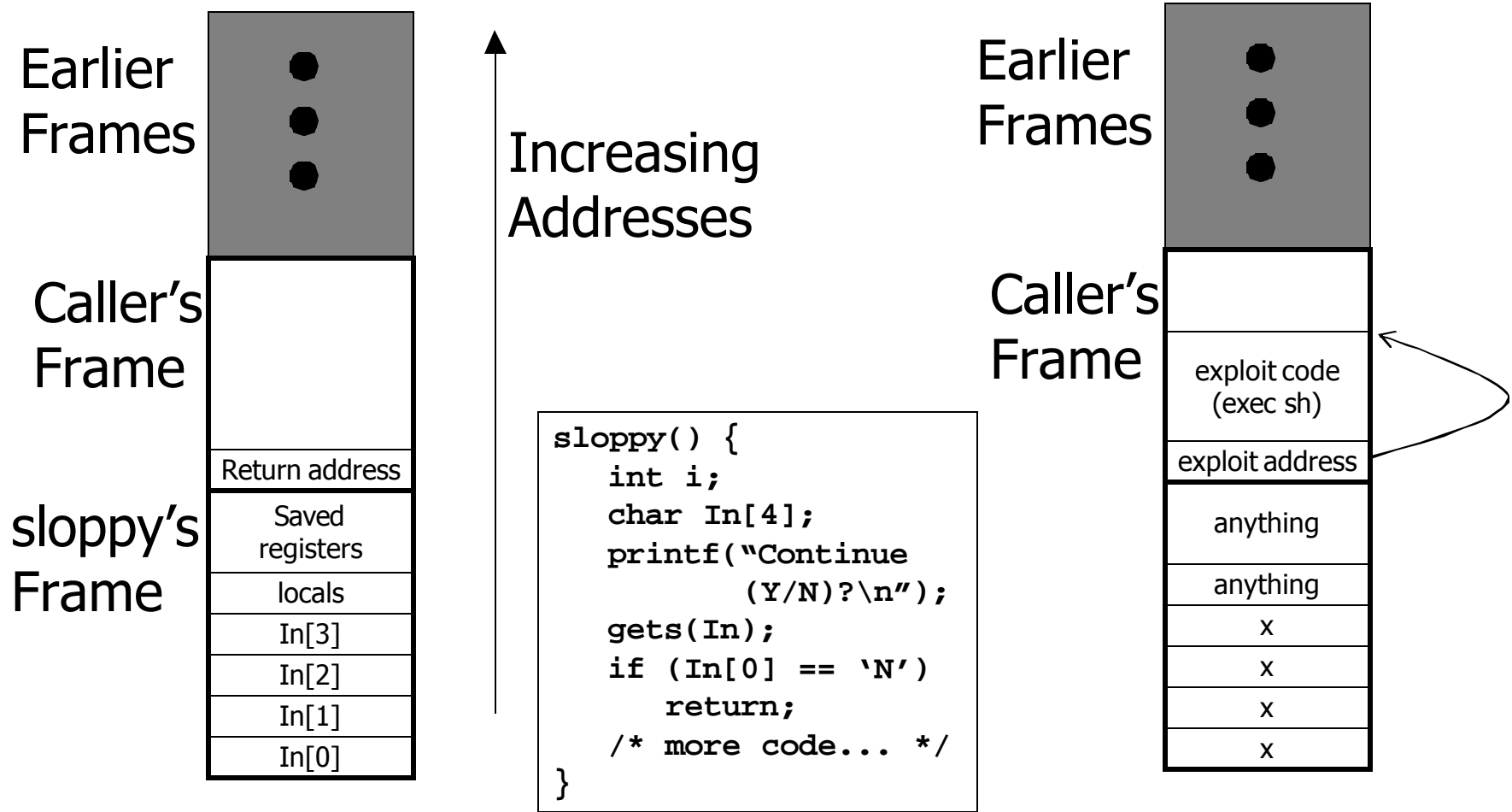
Malicious Software

- Trap doors - program contains secret entry point that allows attacker to bypass security
- Logic bombs - destructive code in legitimate program triggered by some event
- Trojan horses - apparently useful program that tricks users into running it.
 - May contain a logic bomb, or be a vehicle for spreading a virus
- Viruses
 - A program that can "infect" other programs by copying itself into them
- Worms
 - Program that spreads via network connections

Stack & Buffer Overflow Attacks

- Most common means of gaining unauthorized access to a system
- How it works:
 - Overflow some stack-allocated input buffer past the space allocated for the input
 - Overwrite the return address with the address of exploit code
 - Overwrite next space in stack with the exploit code itself
- ??? Let's see what this looks like...

Stack Attack



Trusted Computing Base (TCB)

- Think carefully about what you trust with your data
 - If you type your password on a keyboard, you're trusting
 - The keyboard manufacturer
 - Your computer manufacturer
 - Your OS
 - The password library
 - The application that is checking the password
 - TCB = set of components (hardware, software, people) that you trust your secrets with
- Public Web kiosks should **not** be in your TCB
 - Should your OS? (Think about IE and ActiveX)

Security Design Principles

- Security is much, much more than just crypto
 - If there is a fundamental flaw in the design of the system, then all of the crypto in the world won't help you
 - It is usually easier to find a bug in an implementation than circumvent a crypto systems
- Unfortunately, systems design is still as much an art as it is a science
 - But, decades of building systems the wrong way have helped us gain some learned wisdom
 - We'll cover some in the rest of this lecture

Principle of Least Privilege

- Figure out exactly which capabilities a program needs to run, and grant it only those
 - Not always easy, but one algorithm: start with granting none, run and see where it breaks, add new privileges, repeat
- Unix
 - Good example: Should not normally run as root to guard against accidents
 - Bad example: Some programs run as root just to get a small privilege, such as using a port < 1024 (privileged port)
 - E.g., ftpd, httpd
 - Exploit these programs, and you get root access to system

Principle of Least-Common Mechanism

- Basic lesson: Be careful of shared code
 - Assumptions made may no longer be valid with shared code
- Eudora/Outlook and Internet Explorer
 - Windows exports an API to IE's HTML rendering code
 - Eudora and other programs use this to display HTML in email
 - By default, JavaScript and Java parsing are enabled
 - HTML rendering code knows Java(Script) is unsafe
 - Disables it when JavaScript is downloaded from Internet
 - Internet is untrusted
 - But enables it when JavaScript is loaded off of disk
 - Your own file system is trusted
 - But...email is loaded off of disk!
 - Fertile ground for email viruses...

Principle of Complete Mediation

- Check every access to every object
 - In rare cases, can get away with less (caching)
 - But only if sure nothing relevant in environment has changed (which is a lot)
- Ex: NFS and file handles
 - NFS protocol
 - Client contacts remote "mountd" to get a file handle to a remotely exported NFS file system
 - Remote mountd checks access control at mount time
 - File handle is a capability: client presents it to read/write file
 - Access control is not checked after mount time
 - Can use network sniffer to get file handle and access file system

Consequence: TOCTOU bugs

- NFS example is one instance of a "time-of-check-to-time-of-use" bug / security flaw

"access" checks filename for access permission using real user and group id, not effective id.

... of race condition
... following code fragment
... with setuid root

"open" uses effective user id (root in this case) to open the file.

```
if (access(filename, W_OK) == 0) {  
    if ((fd = open(filename, O_WRONLY)) == NULL) {  
        perror(filename);  
        return 0;  
    }  
    /* Now write to the  
}
```

Attacker:

```
% touch /tmp/foo  
% run_priv /tmp/foo  
% rm /tmp/foo  
% ln /etc/passwd /tmp/foo
```

Fail-Safe Defaults

- Start by denying all access, then allow only that which has been explicitly permitted
 - Oversights will then show up as "false negatives"
 - Somebody is denied access who should have it
 - Opposites lead to "false positives"
 - Somebody is given access that shouldn't get it
 - Not much incentive to report this kind of failure...
- Examples
 - SunOS shipped with "+" in /etc/hosts.equiv
 - Essentially lets anyone login as any local user to host
 - Irix shipped with "xhost +"
 - Any remote client can connect to local X server

No Security Through Obscurity

- Security through obscurity
 - Attempting to gain security by hiding implementation details
 - Claim: A secure system should be secure even if all implementation details are published
 - In fact, systems become more secure as people scour over implementation details and find flaws
 - Rely on mathematics and sound design to provide security
- Ex: GSM cell phones
 - GSM committee designed their own crypto algorithm, but hid it from the world
 - Social engineering + reverse engineering revealed the algorithm
 - Turned out to be relatively weak, easy to subvert

Outlook For The Future

- Doesn't look bright...
 - More and more complex systems are being deployed
 - More and more lives are being trusted to them
- Bruce Schneier: 3 waves of security attacks
 - 1st wave: physical attacks on wires and hardware
 - Physical security to defend against this
 - 2nd wave: syntactic attacks on crypto protocols and systems
 - E.g., buffer overflows, DDoS attacks
 - 3rd wave: semantic attacks: humans and computers trust information that they shouldn't
 - E.g., falsified press announcements
 - Emulex corp stock hoax: CEO "resigns", 61% stock drop
 - Semantic attack against people with preprogrammed sell orders

Case Study: SSL

- Recall crypto basics
 - Messages sent over insecure networks are encoded using a secret key
 - Only parties with the correct key can decode the message
 - Two main types of keys
 - Symmetric, or private key, cryptography
 - Same key is used to encode and decode
 - Encode/decode operations are computationally fast
 - Asymmetric, public key, cryptography
 - 2 keys are needed, one is kept secret, the other is public
 - Messages encoded with one key can be decoded with the other
 - Enables both privacy and authentication
 - Encode/decode operations are computationally expensive

Secure Sockets Layer

- SSL == Secure Sockets Layer
 - Used for secure communications
 - Typically seen in web services (https://)
 - Can be used for any service above TCP/IP layer
- Uses both public key and private key cryptography
- Communication begins with a **handshake protocol** between client and server to establish identity and set up **session keys** used to encrypt remainder of the transmissions
 - Requires the existence of a trusted **Certification Authority (CA)**
 - CA issues certificate to verify server's identity (i.e., its name, IP address, and public key), signed with CA's private key
 - Client can verify certificate (and thus server identity) using CA's public key
 - Web applications ship with a (long) list of CA's pre-installed

SSL Handshake Protocol (1)



ClientHello(version, random_c, session id, ciphers)

ServerHello(version, random_s, session id, ciphers)

ServerCertificate(certificate list)

ServerHelloDone()

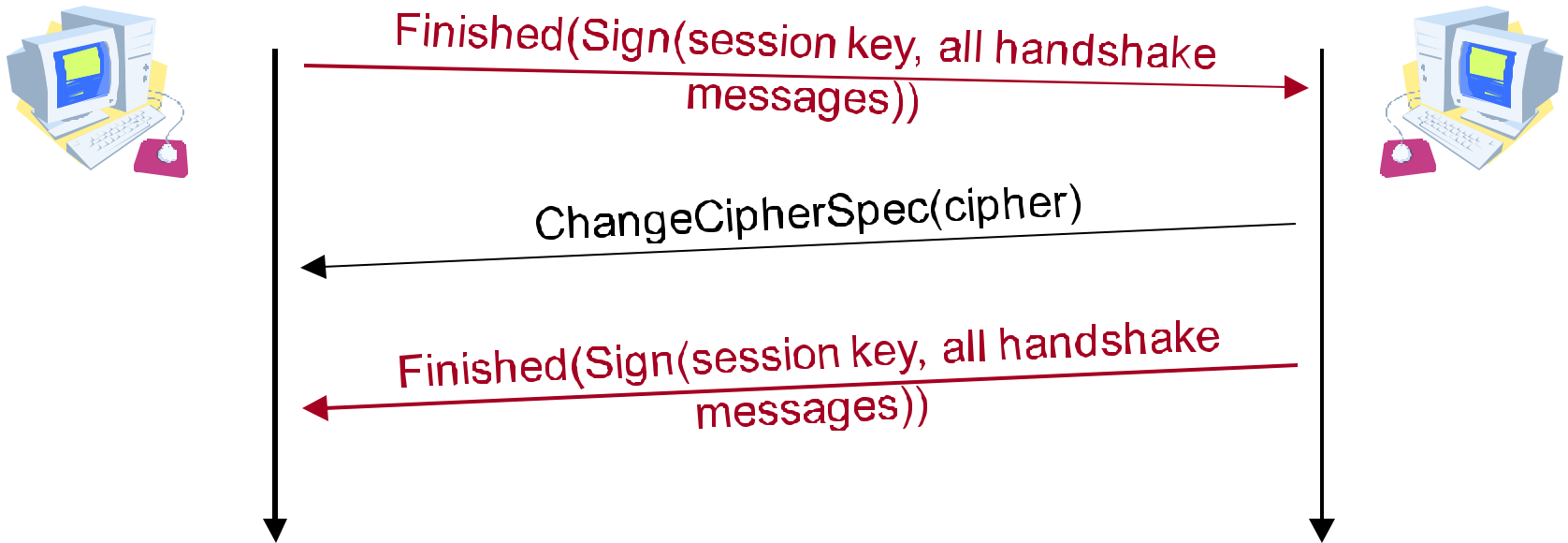
Client verifies server identity.
Generates random pre-master secret.
Selects cipher.

ClientKeyExchange(E(server public key, pre-master secret))

ChangeCipherSpec(cipher)

Now client & server each generate shared secret session keys using selected cipher and (random_c, random_s, pre-master secret)

SSL Handshake Protocol (2)



All future communication is encrypted using session key
("Finished" messages are encrypted and signed)

Notes on SSL Handshake Protocol

- Server certificate signed by certification authority prevents masquerade and man-in-the-middle attacks
 - No masquerade: imposter can present server's real certificate, but does not have server's private key, so can't really pretend to be server. Imposter can't present own certificate pretending to be server unless CA is compromised
 - Man-in-the-middle is a double masquerade (attacker plays role of client to server and plays role of server to client).
- Using random numbers proposed by both client and server in secret key generation prevents replay attacks
 - No server random: attacker can replay client handshake messages to server; server believes it is talking to original client
 - No client random: attacker can replay server's half of session resume handshake
- Using just client/server random values would allow any eavesdropper to also calculate the session keys, since they are sent in the clear. The pre-master secret means only client and server can calculate keys