Lecture 25: Security

Instructor: Mitch Neilsen

Office: N219D

Quote of the Day

“Technology is like water; it wants to find its level. So if you hook up your computer to a billion other computers, it just makes sense that a tremendous share of the resources you want to use - not only text or media but processing power too - will be located remotely. ”

-- Marc Andreessen

Chapter 15: Security

- The Security Problem
- Program Threats
- System and Network Threats
- **Cryptography as a Security Tool**
- User Authentication
- **Implementing Security Defenses**
- **Firewalls to Protect Systems and Networks**

Types of Cryptographic Algorithms

- **Hashing algorithms -** used for authentication and digital signatures.
- **Encryption algorithms -** used for secure voice and data transmission.

RSA Algorithm

- Choose two large prime numbers *p* and *q*.
- Multiply *p* and *q* together to get *n*.
- Choose an encryption key *e*, such that *e* and Euler's totient function $\mathbf{\Phi(n) = (\mathit{p} - 1) * (\mathit{q} - 1)}$ are relatively prime. Two numbers are relatively prime if they have no common factor greater than one.
- Compute decryption key *d* such that

$$d = e^{-1} \bmod ((p - 1) * (q - 1))$$

$$d * e \bmod ((p - 1)*(q - 1)) = 1$$

- Construct public key K_U as (***e, n***).
- Construct private key K_R as (***d, n***).
- Discard (do not disclose) original primes *p* and *q*.

Example

Let p=7 and q=17; so, n=119.

Calculate $\Phi(n) = (p-1)*(q-1) = 6*16 = 96$.

Select e = 5 (note gcd(5,96)=1).

Determine d = 77. Note:5*77=385=4*96+1.

$K_U = (5,119)$, $K_R = (77,119)$.

Now, encrypt $m = 19$.

$$\boldsymbol{c = m^{e} \bmod n}$$

$$\boldsymbol{m = c^{d} \bmod n}$$

Since e = 5, compute $m^{5} = 19^5 = 2{,}476{,}099$; and 2,476,099 mod 119 = 66. So, c = 66.

To decrypt, compute $m = c^{77} \bmod 119 = 19$.

$66^{77} \bmod 119$

$66^2 = 66*66 = 4356$, $4356 \bmod 119 = 72$

$66^4 \bmod 119 = 72^2 \bmod 119 = 5184 \bmod 119 = 67$

$66^8 \bmod 119 = 67^2 \bmod 119 = 4489 \bmod 119 = 86$

$66^{16} \bmod 119 = 86^2 \bmod 119 = 7396 \bmod 119 = 18$

$66^{32} \bmod 119 = 18^2 \bmod 119 = 324 \bmod 119 = 86$

$66^{64} \bmod 119 = 86^2 \bmod 119 = 18$

$$
\begin{aligned}
66^{77} \bmod 119 &= 66^{64} * 66^8 * 66^4 * 66^1 \bmod 119 \\
&= 18 * 86 * 67 * 66 \bmod 119 \\
&= 6{,}845{,}256 \bmod 119 \\
&= 19
\end{aligned}
$$

Problem

Let p=11 and q=13; so, n=143.

Calculate $\Phi(n) = (p-1)*(q-1) = 10*12 = 120 = 2^3 * 3 * 5$.

Select e = 7 (note gcd(7, 120) = 1).

Determine d =

$K_U = (7, 143)$, $K_R = ($ __ $, 143)$.

Now, encrypt $m = 25$.

$$c = m^e \bmod n$$

$$m = c^d \bmod n$$

Since e = 3, compute $m^7 = 25^7 = 6{,}103{,}515{,}625$; and 6,103,515,625 mod 143 = 64. So, c = 64.

To decrypt, compute $m = c^{—} \bmod 143 =$ __ .

Problem

Let p=11 and q=13; so, n=143.

Calculate $\Phi(n) = (p-1)*(q-1) = 10*12 = 120 = 2^3 * 3 * 5$.

Select e = 7 (note gcd(7, 120) = 1).

Determine d = **103**

$K_U = (7, 143)$, $K_R = (\mathbf{103}, 143)$.

Now, encrypt *m* = 25.

$$\boldsymbol{c = m^{e} \bmod n}$$

$$\boldsymbol{m = c^{d} \bmod n}$$

Since e = 3, compute $m^{7} = 25^7 = 6{,}103{,}515{,}625$; and 6,103,515,625 mod 143 = 64. So, c = 64.

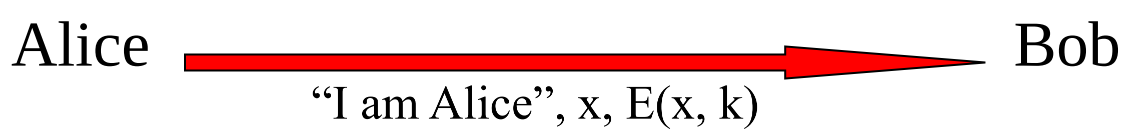
To decrypt, compute $m = c^{\mathbf{103}} \bmod 143 = 64^{\mathbf{103}} \bmod 143 = 25$.

Secret Key Authentication

Alice wants to talk to Bob

- Needs to convince him of her identity
- Both have same single (secret) key k

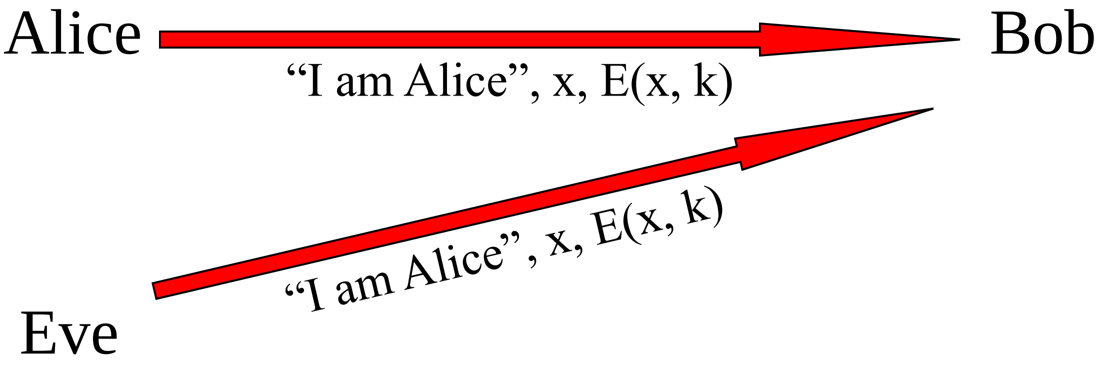
Naive scheme



Vulnerability?

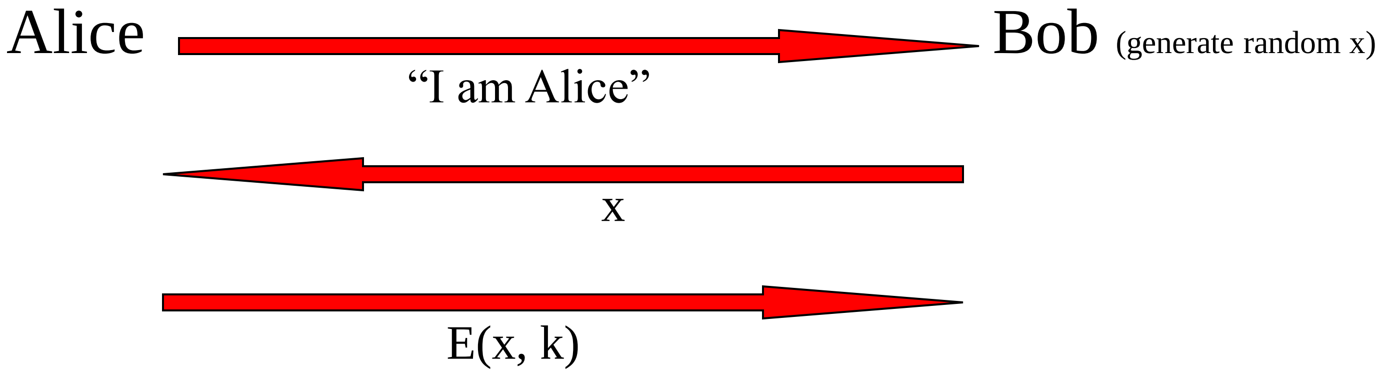
Replay Attack

Eve can listen in and impersonate Alice later



Preventing Replay Attacks

Bob can issue a challenge phrase to Alice

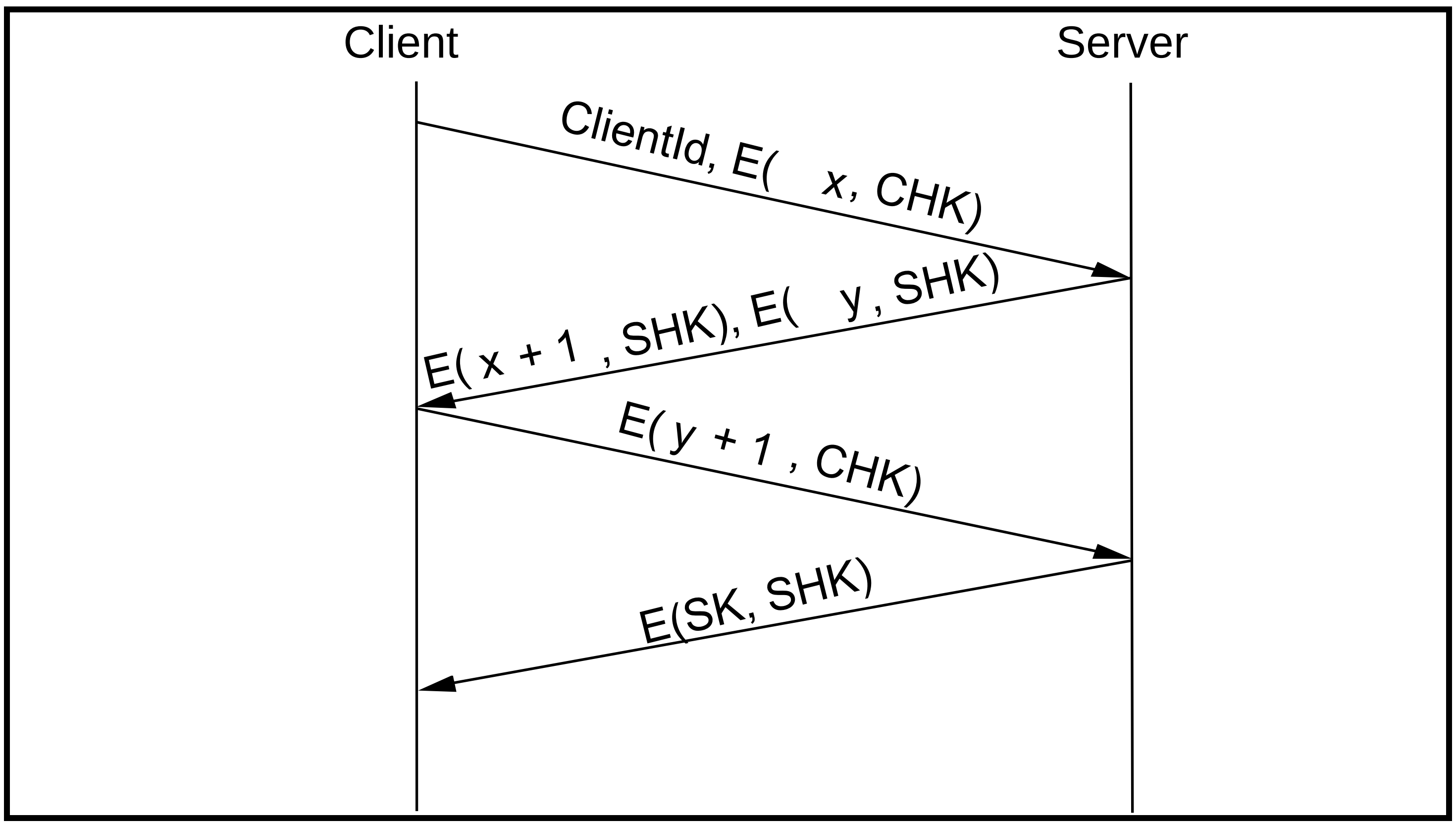


Authentication Protocols

Three-way handshake: E(msg, key) – encrypt msg with key

CHK = Client handshake key,   SHK = server handshake key

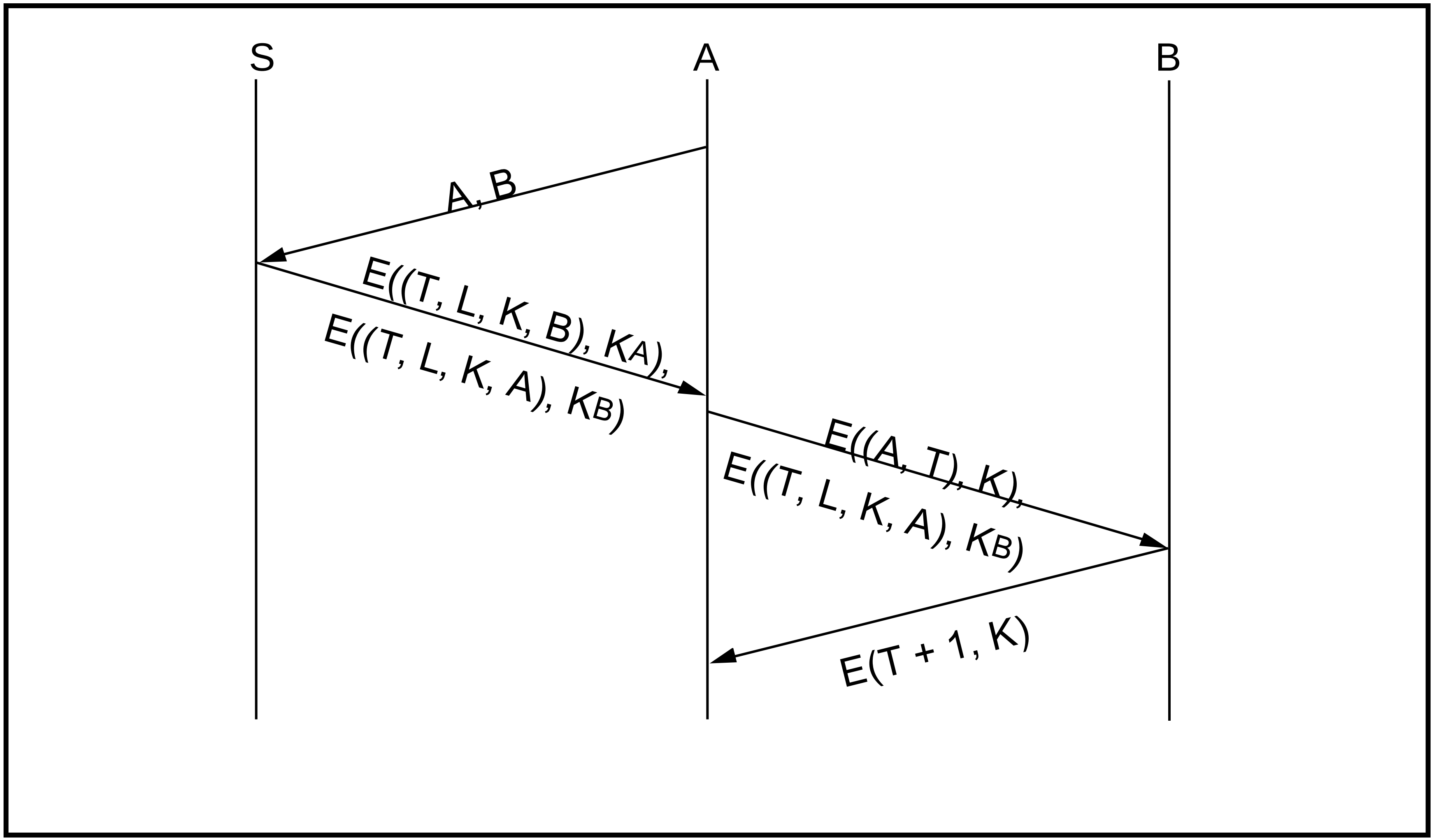
SK = Secret session key



Authentication Protocols (cont.)

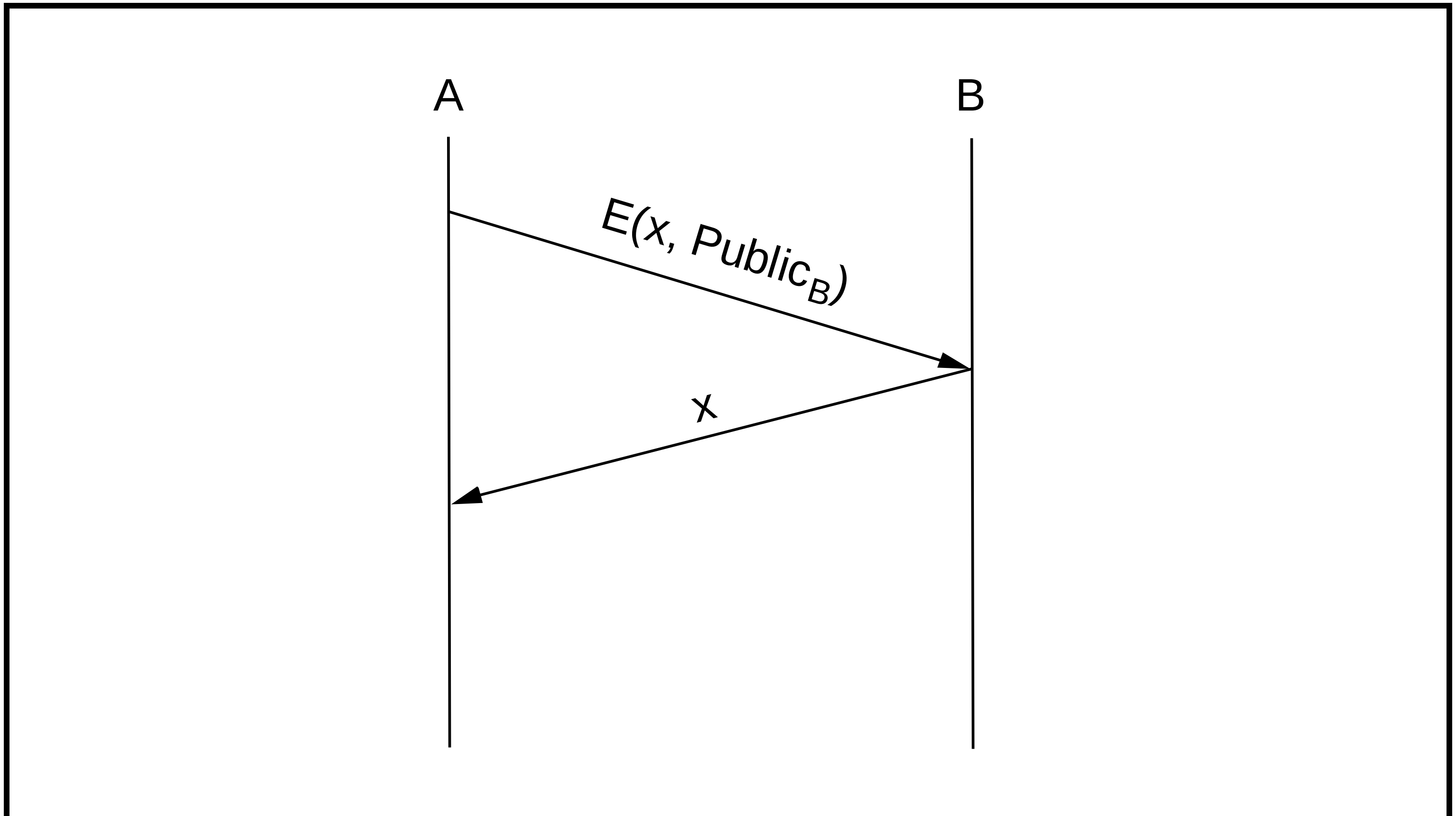
Trusted third party: S = authentication server using Kerberos

T = timestamp, L = lifetime, K = session key



Public Key Authentication

x = random value, $Public_B$ = B's public key
B authenticates itself by sending x back to A



Cryptographic Hash Functions

Given arbitrary length m, compute constant length digest d = h(m)

Desirable properties

- h(m) easy to compute given m
- One-way: given h(m), hard to find m
- Weakly collision free: given h(m) and m, hard to find m’ s.t. h(m) = h(m’)
- Strongly collision free: hard to find any x, y s.t. h(x) = h(y)

Example use: password database, file distribution, digital signature

Common algorithms: MD5, SHA-1, RIPEMD-160

Message Digest (MD5)

Cryptographic checksum

- just as a regular checksum protects the receiver from accidental changes to the message, a cryptographic checksum protects the receiver from malicious changes to the message.

One-way function

- given a cryptographic checksum for a message, it is virtually impossible to figure out what message produced that checksum; it is not computationally feasible to find two messages that hash to the same cryptographic checksum.

Relevance

- if you are given a checksum for a message and you are able to compute exactly the same checksum for that message, then it is highly likely this message produced the checksum you were given.

MD5

- MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit words); the message is padded so that its length is divisible by 512.
- The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words, denoted *A*, *B*, *C* and *D*. These are initialized to certain fixed constants. The main algorithm then uses each 512-bit message block in turn to modify the state. The processing of a message block consists of four similar stages, termed *rounds*; each round is composed of 16 similar operations based on a non-linear function *F*, modular addition, and left rotation.

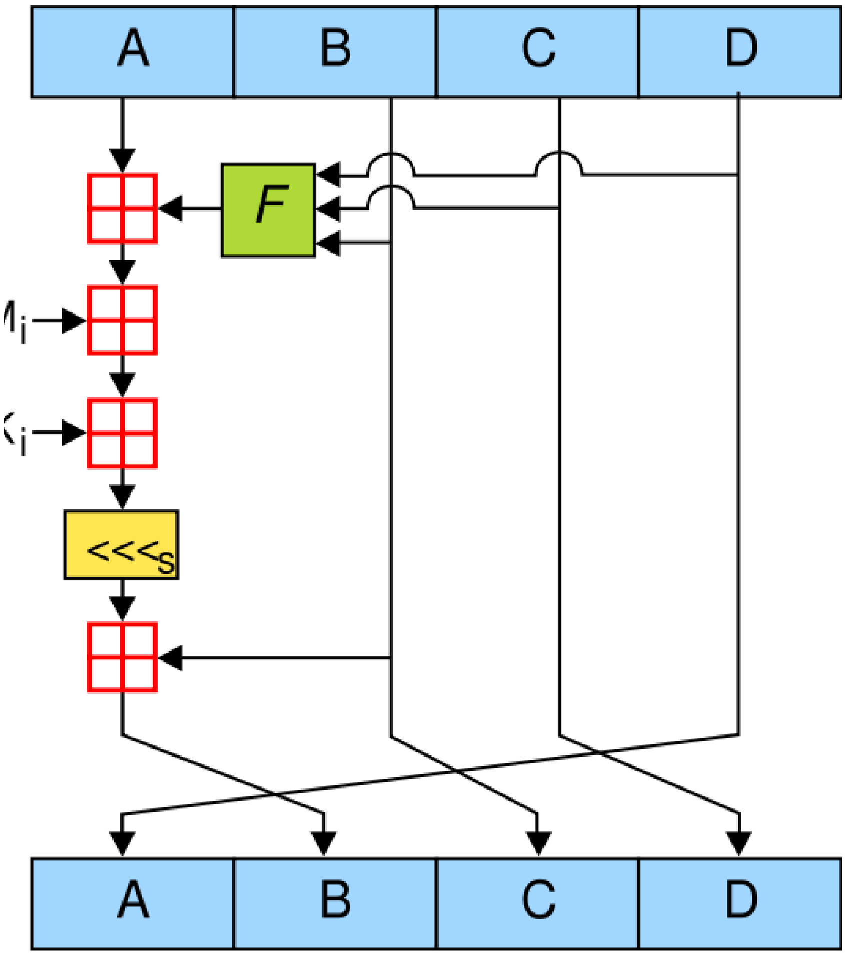
--wikipedia

MD5

Four different functions, F,G,H,I, can be used in each round:

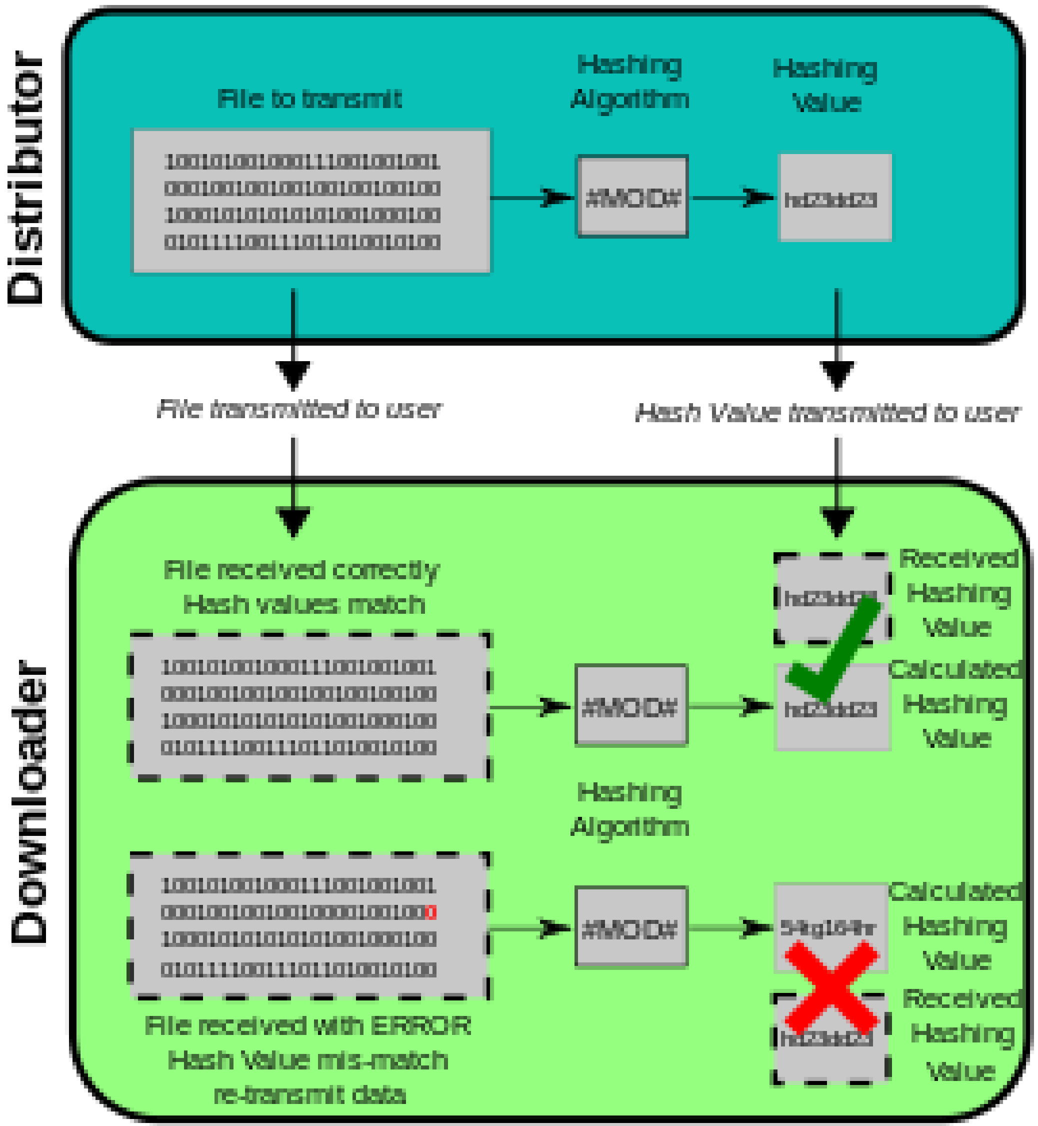
$$F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$$
$$G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$$
$$H(B,C,D) = B \oplus C \oplus D$$
$$I(B,C,D) = C \oplus (B \vee \neg D)$$

$\oplus, \wedge, \vee, \neg$ denote the XOR, AND, OR and NOT operations respectively.



--wikipedia

Secure File Transmission



--wikipedia

Message Integrity Protocols

Digital signature using RSA

- special case of a message integrity where the code can only have been generated by one participant
- compute signature with private key and verify with public key

Keyed MD5

- sender: $\boldsymbol{m}$ **+ MD5(**$\boldsymbol{m}$ **+** $\boldsymbol{k}$ **) + E(**$\boldsymbol{k}$**,** $\boldsymbol{k_{private}}$**)**
- receiver

  recovers random key $\boldsymbol{k}$ using the sender's public key $\boldsymbol{k_{public}}$

  applies MD5 to the concatenation of this random key with the message

MD5 with RSA signature

- sender: $\boldsymbol{m}$ **+ E(MD5(**$\boldsymbol{m}$**),** $\boldsymbol{k_{private}}$**)**
- receiver

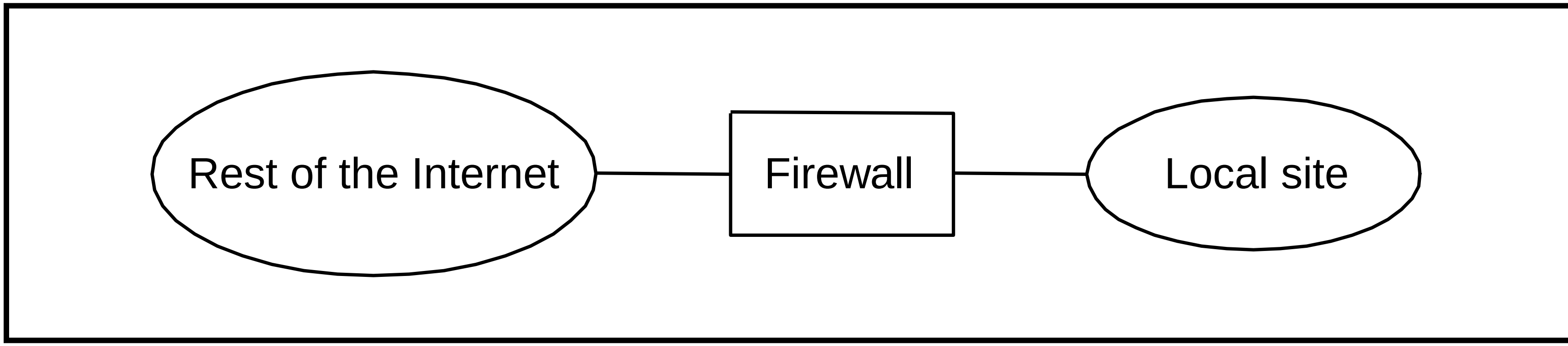
  decrypts signature with sender's public key

  compares result with MD5 checksum sent with message

Firewalls

- Basic problem – many network applications and protocols have security problems that are fixed over time
  - Difficult for users to keep up with changes and keep host secure
  - Solution
    - Administrators limit access to end hosts by using a firewall
    - Firewall and limited number of machines at site are kept up-to-date by administrators

Firewalls



Filter-Based Solution

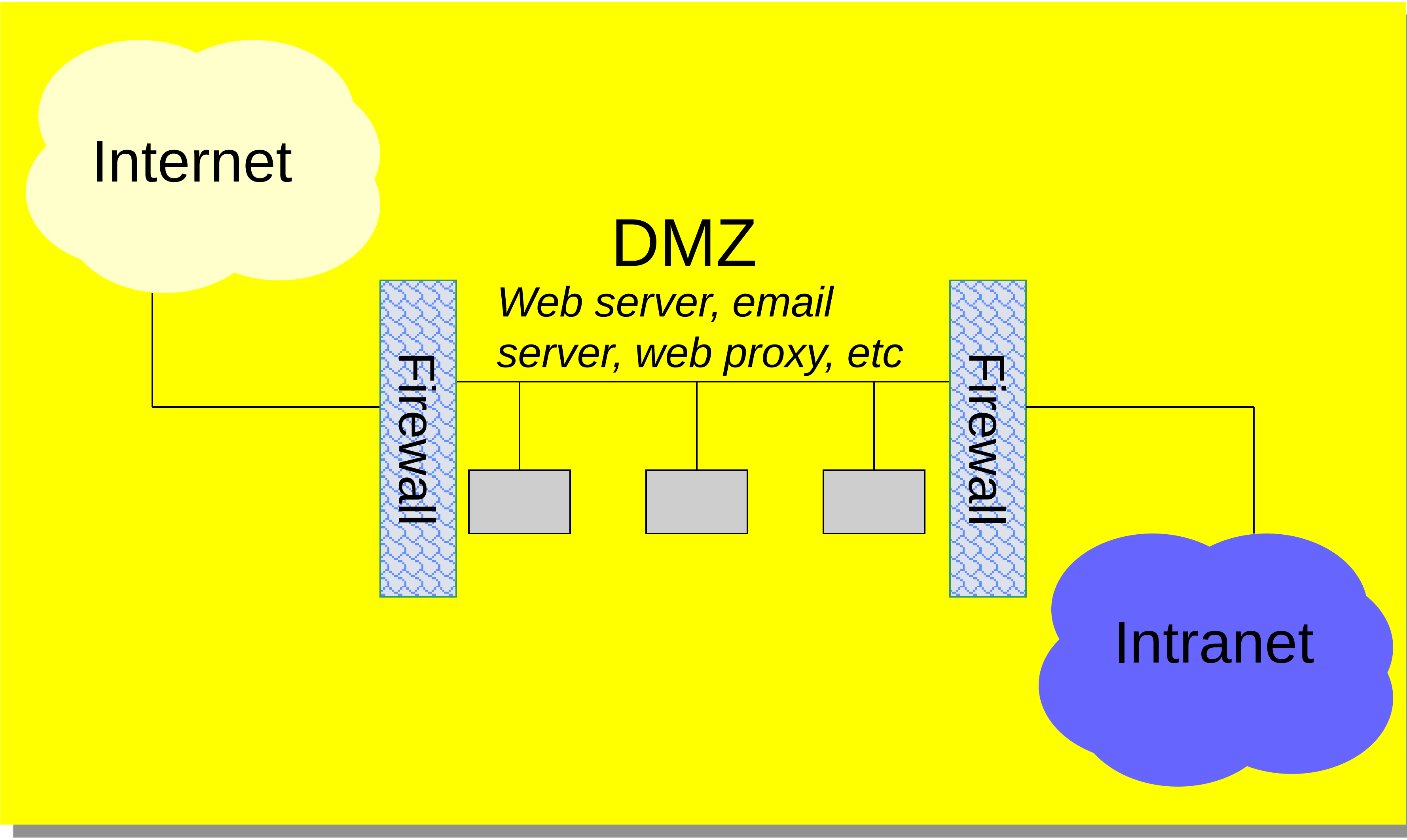
- example

  (128.130.10.5, 1234, 129.130.10.16, 80)

  (*,*, 129.130.10.16, 80)

- default: forward or not forward?
- how dynamic?

Typical Firewall Topology



Access Control

- **Discretionary Access Control -** restricting access to objects based on the identity of subjects and/or groups to which they belong; e.g., Unix perms. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (either directly or indirectly) on to any other subject; e.g., chmod.
- **Mandatory Access Control -** the operating system constrains access to objects.

DAC vs. MAC

- **Most people familiar with discretionary access control (DAC)**
  - Unix permission bits are an example
  - Might set a file permissions so that only group 'friends' can read it
- **Discretionary means anyone with access can propagate information:**
  - Mail sigint@enemy.gov < private
- **Mandatory Access Control (MAC)**
  - Security administrator can restrict propagation
  - Abbreviated MAC (NOT to be confused w. Message Authentication Code or Medium Access Control)

Bell-Lapadula model

- **View the system as subjects accessing objects**
  - The system input is requests, the output is decisions
  - Objects can be organized in one or more hierarchies, *H* (a tree enforcing the type of descendents)
- **Four modes of access are possible:**
  - execute – no observation or alteration
  - read – observation
  - append – alteration
  - write – both observation and modification
- **The current access set, *b*, is (subj, obj, attr) triples**
- **An access matrix *M* encodes permissible access types (as before, subjects are rows, objects columns)**

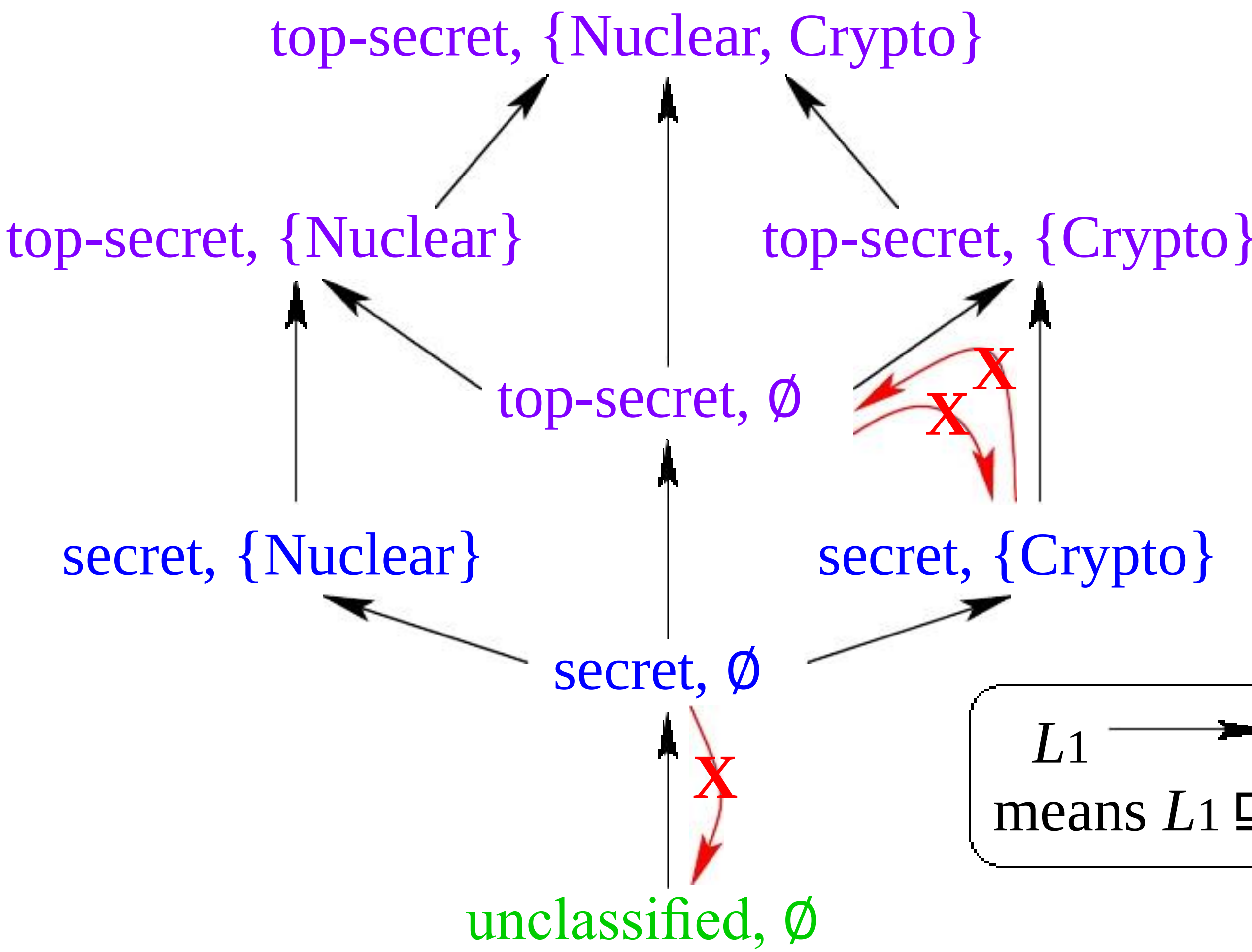
Security levels

- **A *security level* is a** (c, s) **pair:**
  - c = classification – E.g., unclassified, secret, top secret
  - s = category-set – E.g., Nuclear, Crypto
- (c_1, s_1) ***dominates*** (c_2, s_2) **iff** $c_1 \geq c_2$ **and** $s_2 \subseteq s_1$
  - L_1 *dominates* L_2 sometimes written $L_1 \sqsupseteq L_2$ or $L_2 \sqsubseteq L_1$
  - levels then form a *lattice* (partial order w. lub & glb)
- **Subjects and objects are assigned security levels**
  - level(S), level(O) – security level of subject/object
  - current-level(S) – subject may operate at lower level
  - level(S) bounds current-level(S) (current-level(S) $\sqsubseteq$ level(S))
  - Since level(S) is max, sometimes called S's *clearance*

Security properties

- **The simple security or *ss-property*:**
  - For any $(S, O, A) \in b$, if A includes observation, then level(S) must dominate level(O)
  - E.g., an unclassified user cannot read a top-secret document
- **The star security or **-property*:**
  - If a subject can observe O_1 and modify O_2, then level(O_2) dominates level(O_1)
  - E.g., cannot copy top secret file into secret file
  - More precisely, given $(S, O, A) \in b$:
    if $A = r$ then current-level(S) $\sqsupseteq$ level(O) (“no read up”)
    if $A = a$ then current-level(S) $\sqsubseteq$ level(O) (“no write down”)
    if $A = w$ then current-level(S) = level(O)

The lattice model



$L1 \longrightarrow L1$
means $L1 \sqsubseteq L2$

- **Information can only flow up the lattice**
  - System enforces "No read up, no write down"
  - Think of $\sqsubseteq$ as "can flow to" relation

Straw-man MAC implementation

- **Take an ordinary Unix system**
- **Put labels on all files and directories to track levels**
- **Each user** U **has a security clearance,** level(U)
- **Determine current security level dynamically**
    - When U logs in, start with lowest curent-level
    - Increase current-level as higher-level files are observed (sometimes called a *floating label* system)
    - If U's level does not dominate current-level, kill program
    - Kill program that writes to file that doesn't dominate it
- **Is this secure?**

No: Covert channels

- **System rife with *storage channels***
  - Low current-level process executes another program
  - New program reads sensitive file, gets high current-level
  - High program exploits covert channels to pass data to low
- **E.g., High program inherits file descriptor**
  - Can pass 4-bytes of information to low prog. in file offset
- **Other storage channels:**
  - Exit value, signals, file locks, terminal escape codes, . . .
- **If we eliminate storage channels, is system secure?**

No: Timing channels

- **Example: CPU utilization**
  - To send a 0 bit, use 100% of CPU in busy-loop
  - To send a 1 bit, sleep and relinquish CPU
  - Repeat to transfer more bits
- **Example: Resource exhaustion**
  - High prog. allocates all physical memory if bit is 1
  - If low prog. slow from paging, knows less memory available
- **More examples: Disk head position, processor cache/TLB polution, . . .**

Reducing covert channels

- **Observation: Covert channels come from sharing**
  - If you have no shared resources, no covert channels
  - Extreme example: Just use two computers (common in DoD)
- **Problem: Sharing needed**
  - E.g., read unclassified data when preparing classified
- **Approach: Strict partitioning of resources**
  - Strictly partition and schedule resources between levels
  - Occasionally reapportion resources based on usage
  - Do so infrequently to bound leaked information
  - In general, only hope to bound bandwidth of covert channels
  - Approach still not so good if many security levels possible

Declassification

- **Sometimes need to prepare unclassified report from classified data**
- **Declassification happens outside of system**
  - Present file to security officer for downgrade
- **Job of declassification often not trivial**
  - E.g., Microsoft word saves a lot of undo information
  - This might be all the secret stuff you cut from document
  - Another bad mistake: Redacted PDF using black censor bars over or under text (but text still selectable)

Biba integrity model

- **Problem: How to protect integrity**
  - Suppose text editor gets trojaned, subtly modifies files, might mess up attack plans
- **Observation: Integrity is the converse of secrecy**
  - In secrecy, want to avoid writing less secret files
  - In integrity, want to avoid writing higher-integrity files
- **Use integrity hierarchy parallel to secrecy one**
  - Now *security level* is a c, i, s triple, i =integrity
  - $c_1, i_1, s_1 \sqsubseteq c_2, i_2, s_2$ iff $c_1 \leq c_2$ and $i_1 \geq i_2$ and $s_1 \subseteq s_2$
  - Only trusted users can operate at low integrity levels
  - If you read less authentic data, your current integrity level gets lowered (putting you up higher in the lattice), and you can no longer write higher-integrity files

DoD Orange Book

- **DoD requirements for certification of secure systems**
- **Four Divisions:**
  - D – been through certification and not secure
  - C – discretionary access control
  - B – mandatory access control
  - A – like B, but better verified design
  - Classes within divisions increasing level of security

Limitations of Orange book

- **How to deal with floppy disks, removable storage?**
- **How to deal with networking?**
- **Takes too long to certify a system**
  - People don’t want to run *n*-year-old software
- **Doesn’t fit non-military models very well**
- **What if you want high assurance & DAC?**

Today: Common Criteria

- **Replaced orange book around 1998**
- **Three parts to CC:**
  - CC Documents, including protection profiles with both functional and assurance requirements
  - CC Evaluation Methodology
  - National Schemes (local ways of doing evaluation)

Summary

- Read Ch. 14-15
- Project #3 – new due date, Mon., Dec. 16
  - Sensor Input Application
  - Kernel modifications – add new system call