CSCI 4250/6250 – Fall 2015
Computer and Networks Security

CHAPTER 1 - INTRODUCTION

Research in Computer Security

- Studies in what ways **security mechanisms may fail**

  - Can we gain access to a computer system without authorization?
  - Can we compromise CIA of data?

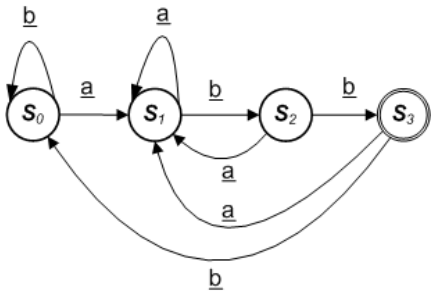
- Understanding the vulnerabilities of a system to develop **better defenses**
  - Secure OSs (only allow authorized use)
  - Secure applications and communications (e.g., secure online banking)

Defining Security

- The security of a system, application, or protocol is always relative to
  - A set of desired properties/policies
  - An adversary with specific capabilities
- For example, standard file access permissions in Linux and Windows are not effective against an adversary who can boot from a CD

- *A system is secure if it starts from a secure state, and is not allowed to transition to states that are deemed not secure*

A more formal definition…



- Consider a computer system as an FSA
- *Security Policy*
  - A statement that partitions the states of the system into *secure* states and *non-secure states*
- *A system is secure if it starts from a secure state, and is not allowed to transition to states that are deemed not secure (according to the security policies)*

A more formal definition…

- *Security Mechanisms*
  - Entities or procedures that are meant to enforce the security policies

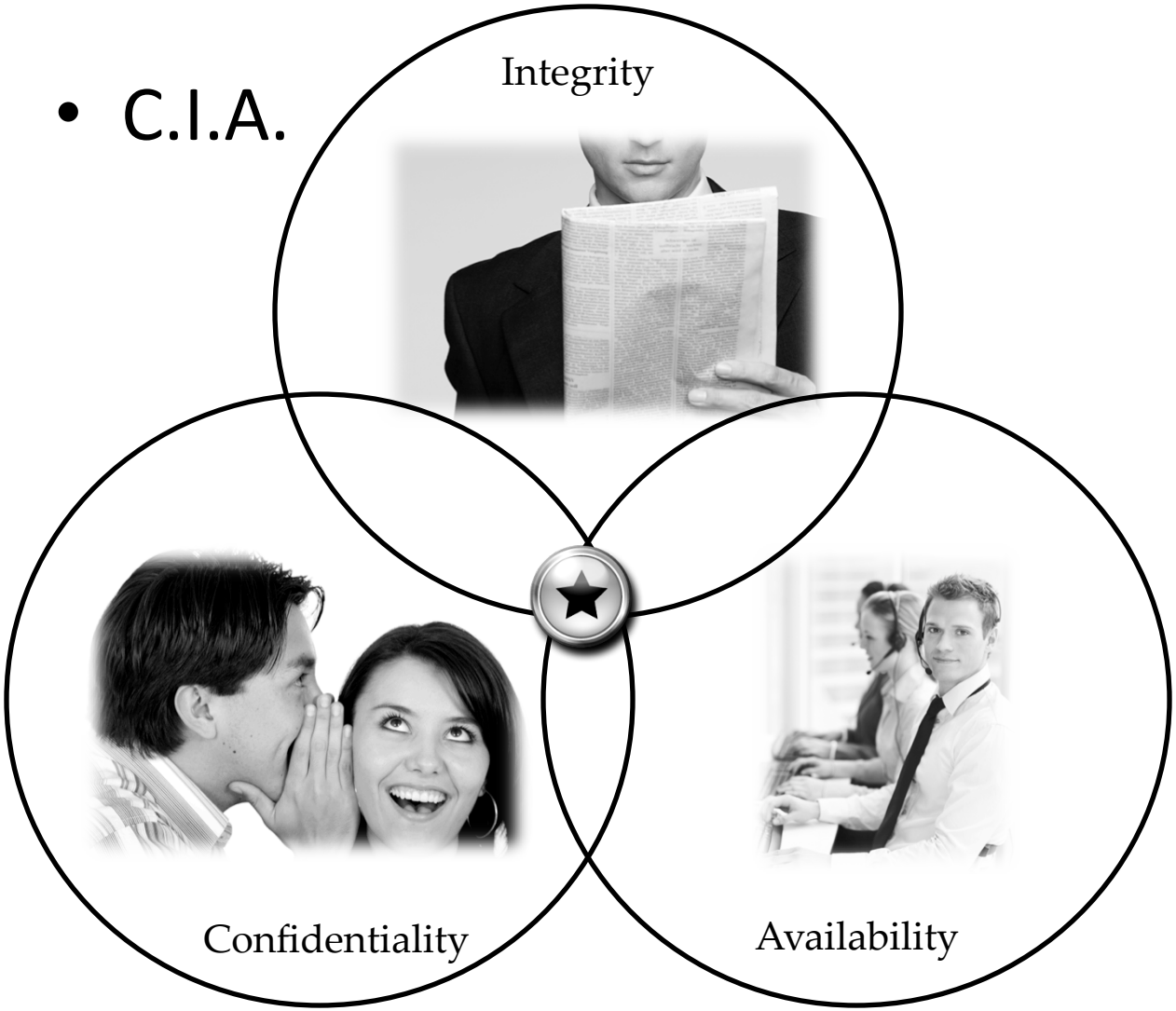
- *A breach of security occurs when a system enters an unauthorized (non-secure) state*
  - *Failure of a security mechanism*

A simple example

- Policy
  - Environment: multi-user computer system
  - Security policy:
    - *a user* U1 *shall not be allowed to delete or modify files belonging to other users, unless the owners of a file explicitly grants such permission to* U1
- Security mechanism:
  - OS file-system access control mechanisms
- Breach of security example:
  - Alice exploits a vulnerability in the OS file-system that allows her to delete other people's files
  - The exploit causes the system to transition from a secure state to a non-secure state

Security Goals

- C.I.A.

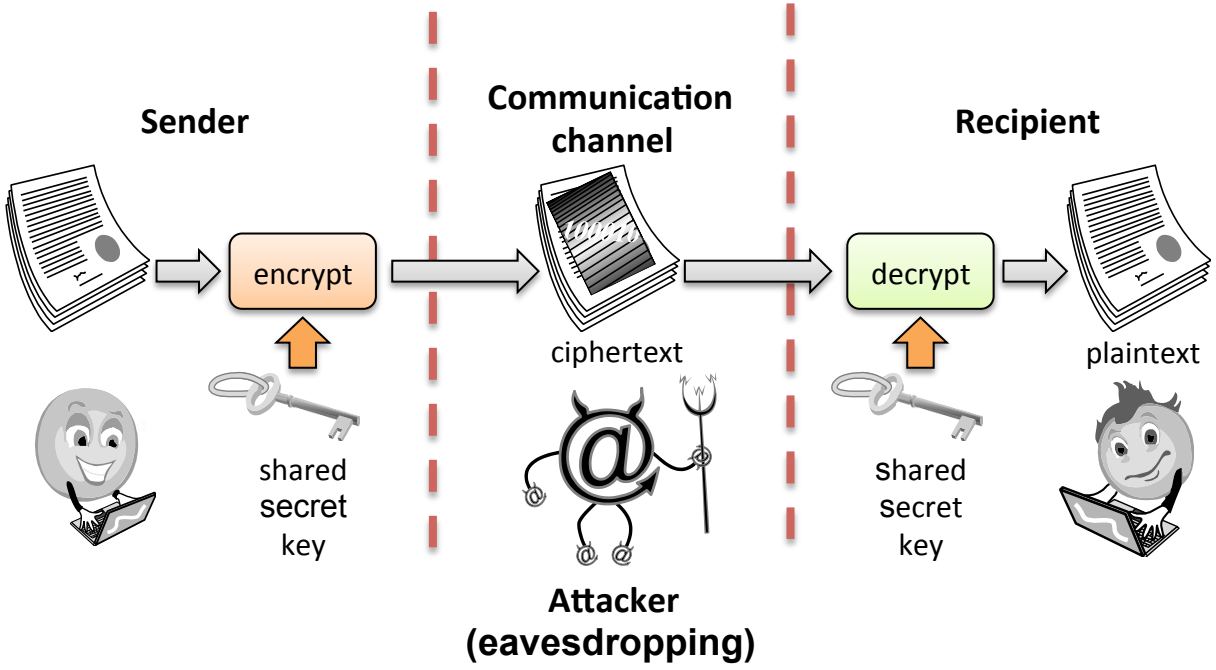


Confidentiality

- **Confidentiality** is the avoidance of the unauthorized disclosure of information.
  - confidentiality involves the protection of data, providing access for those who are allowed to see it while disallowing others from learning anything about its content.

Tools for Confidentiality

- **Encryption:** the transformation of information using a secret, called an encryption key, so that the transformed information can only be read using another secret, called the decryption key (which may, in some cases, be the same as the encryption key).

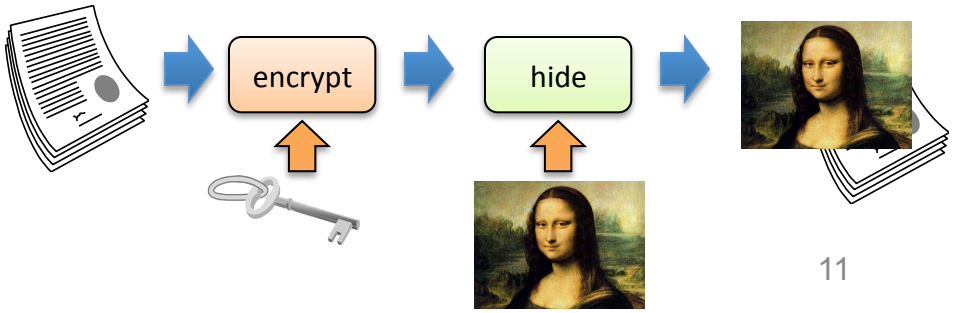


Tools for Confidentiality

- Steganography
  - Conceals the existence of the message
  - If the "location" of the message is found, game over!
- Analogy
  - Hide cash inside a sock in a "unsuspected" drawer chest
  - If a burglar breaks into a villa, the safe will certainly attract attention
  - Break the combination (break the key!)
  - But if they notice the socks full of money, its going to be an easy steel!

Crypto vs. Steganography

- Crypto
  - Garbles the message
  - Encryption algorithm is known, but keys are secret
  - If you send an encrypted message (e.g., email) it may be evident you have something important to hide
- Steganography
  - Based on *security by obscurity*
  - Goal is not to garble the message
  - Plaintext message hidden in some communication that does not attract attention (unless you have some prior knowledge)
- Crypto + Steganography
  - could be easily combined

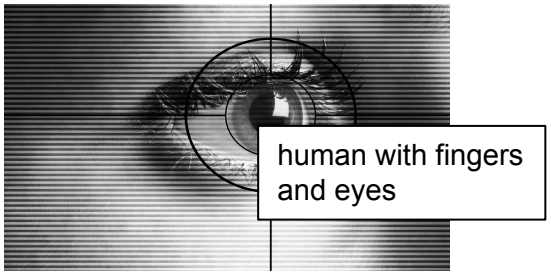


Tools for Confidentiality

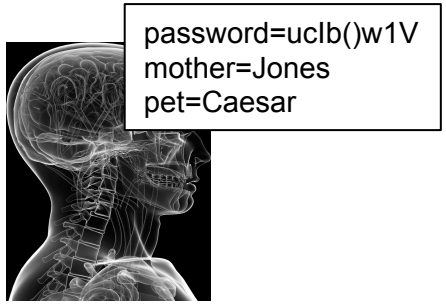
- **Access control:** rules and policies that limit access to confidential information to those people and/or systems with a "need to know."
  - This need to know may be determined by identity, such as a person's name or a computer's serial number, or by a role that a person has, such as being a manager or a computer security specialist.

Tools for Confidentiality

- **Authentication:** the determination of the identity or role that someone has. This determination can be done in a number of different ways, but it is usually based on a combination of
  - something the person has (like a smart card or a radio key fob storing secret keys),
  - something the person knows (like a password),
  - something the person is (like a human with a fingerprint).



Something you are



Something you know

Something you have

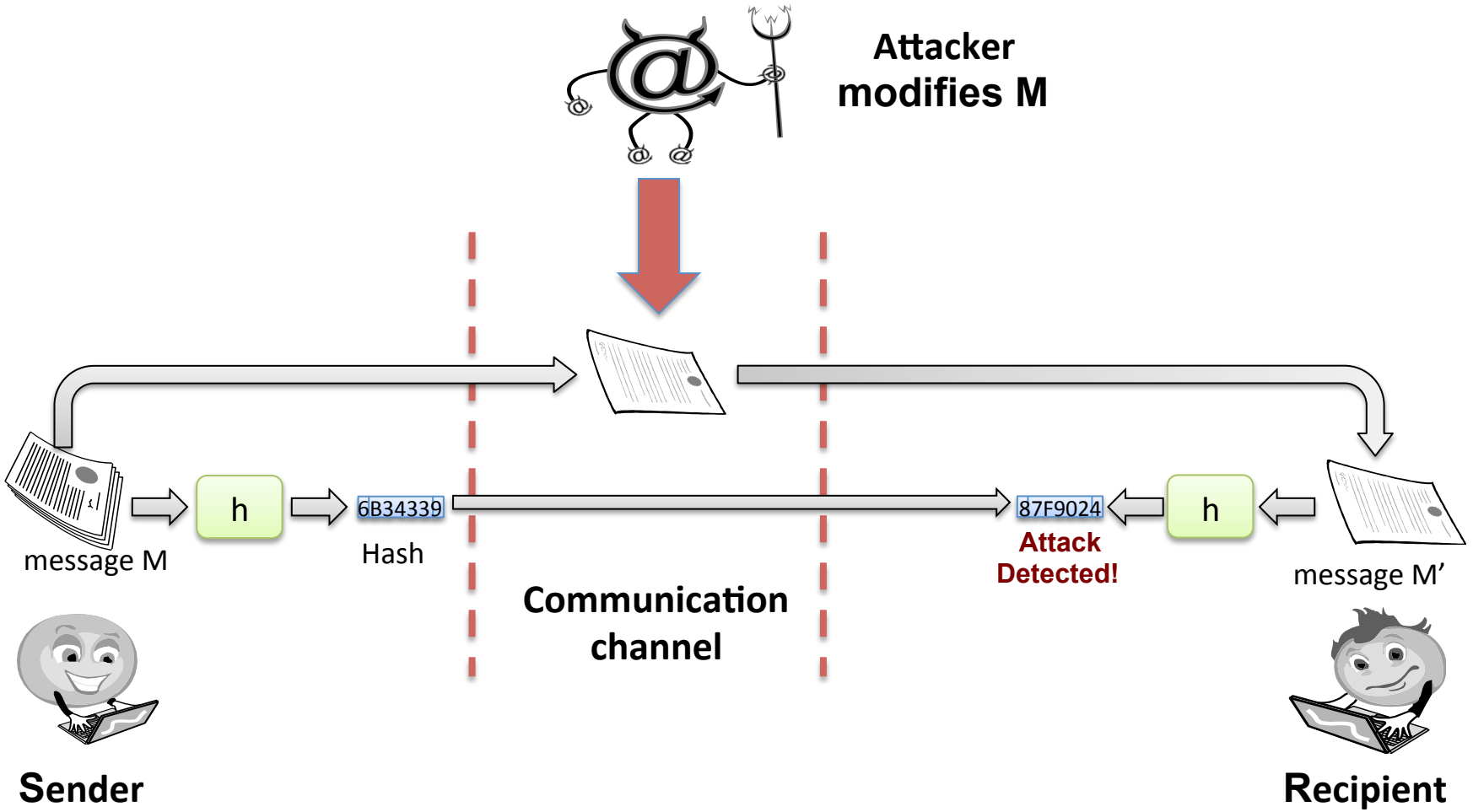
Tools for Confidentiality

- **Authorization:** the determination if a person or system is allowed access to resources, based on an access control policy.
  - Such authorizations should prevent an attacker from tricking the system into letting him have access to protected resources.
- **Physical security:** the establishment of physical barriers to limit access to protected computational resources.
  - Such barriers include locks on cabinets and doors, the placement of computers in windowless rooms, the use of sound dampening materials, and even the construction of buildings or rooms with walls incorporating copper meshes (called **Faraday cages)** so that electromagnetic signals cannot enter or exit the enclosure.

Integrity

- **Integrity:** the property that information has not be altered in an unauthorized way.
- **Tools used to protect integrity:**
  - **Prevention**
    - **Authentication, Authorization**
  - **Detection/Remediation**
    - **Checksums/Hashes:** the computation of a function that maps the contents of a file to a numerical value. A checksum function depends on the entire contents of a file and is designed in a way that even a small change to the input file (such as flipping a single bit) is highly likely to result in a different output value.
    - **Data correcting codes:** methods for storing data in such a way that small changes can be easily detected and automatically corrected.
    - **Backups:** the periodic archiving of data.

Integrity
does this work?



Availability

- **Availability:** the property that information is accessible and modifiable in a timely fashion by those authorized to do so.
- **Tools:**
  - **Physical protections:** infrastructure meant to keep information available even in the event of physical challenges.
  - **Computational redundancies:** computers and storage devices that serve as fallbacks in the case of failures.
  - **Network resources:** traffic monitoring/throttling for DoS detection/mitigation

Other Security Concepts

- A.A.A.

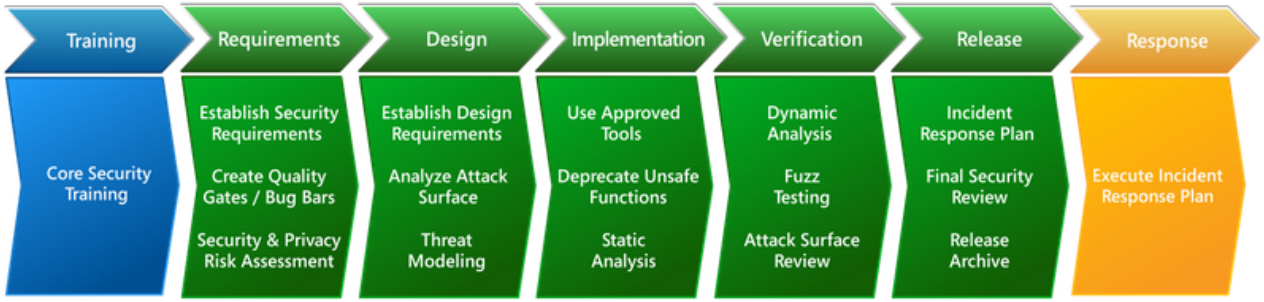
Authenticity

Anonymity

Assurance

Assurance

- **Assurance** refers to how trust is provided and managed in computer systems.
- **Trust management** depends on:
  - **Policies,** which specify behavioral expectations that people or systems have for themselves and others.
    - For example, the designers of an online music system may specify policies that describe how users can access and copy songs.
  - **Permissions,** which describe the behaviors that are allowed by the agents that interact with a person or system.
    - For instance, an online music store may provide permissions for limited access and copying to people who have purchased certain songs.
  - **Protections,** which describe mechanisms put in place to enforce permissions and polices.
    - We could imagine that an online music store would build in protections to prevent people from unauthorized access and copying of its songs.

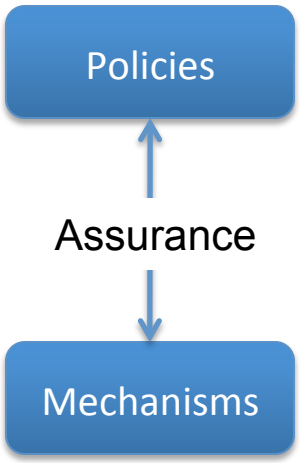


Microsoft Security Development Lifecycle

Assurance
(a more precise definition)

- **Trustworthiness**
  - *An entity is trustworthy if there is sufficient credible evidence leading one to believe that the system will meet a set of given requirements*
- **Security Assurance**
  - Confidence that an entity meets its security requirements (it's trustworthy)
  - Based on specific evidence provided by the application of assurance techniques
    - Secure development methodologies, formal methods for design and analysis, and rigorous testing

Assurance
(a more precise definition)



Statement of requirements
Define security expectations

Provides evidence that mechanisms meet the requirements stated in the policies

Security modules designed and implemented to enforce the policies

- **Trusted System**
  - A system that has been shown to meet well-defined requirements under an evaluation by experts who are certified to evaluate a system and assign trust ratings
  - Experts collect evidence of assurance, and interpret the results to assign level of trustworthiness

The Role of *Trust* in Security

- Security policies and mechanisms rest on a set of assumptions
- Example:
  - You want to improve your security when browsing the Internet
    - Policy: Scripts (e.g., JavaSript) shall never be downloaded, parsed, and executed by the browser
    - Mechanism: you download a “script block” plug-in for your favorite browser

Are you really more secure?

The Role of *Trust* in Security

- Assumptions
  - The plug-in was developed by a trusted vendor and was not tampered with
  - The plug-in will correctly block *all* scripts
  - The plug-in itself does not introduce new vulnerabilities
- What if any of these assumptions is violated?
  - System is not secure
  - Worse yet: false sense of security!

Authenticity

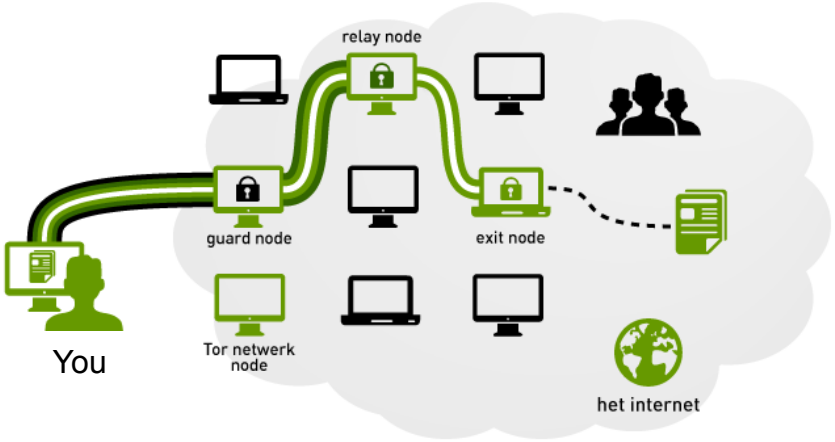
- **Authenticity** is the ability to determine that statements, policies, and permissions issued by persons or systems are genuine.
- **Primary tool:**
  - **digital signatures.** These are cryptographic computations that allow a person or system to commit to the authenticity of their documents in a unique way that achieves **non-repudiation**, which is the property that authentic statements issued by some person or system cannot be denied.

Anonymity

- **Anonymity:** the property that certain records or transactions cannot be attributable to any individual

- **Tools:**
  - **Aggregation:** the combining of data from many individuals so that disclosed sums or averages cannot be tied to any individual.
  - **Mixing:** the intertwining of transactions, information, or communications in a way that cannot be traced to any individual.
  - **Proxies:** trusted agents that are willing to engage in actions for an individual in a way that cannot be traced back to that person.
    - Example: Tor Onion Routing (why do we need to trust the exit nodes?)
  - **Pseudonyms:** fictional identities that can fill in for real identities in communications and transactions, but are otherwise known only to a trusted entity.

- **Examples**
  - Good use: anti-censorship
  - Bad use: attacks



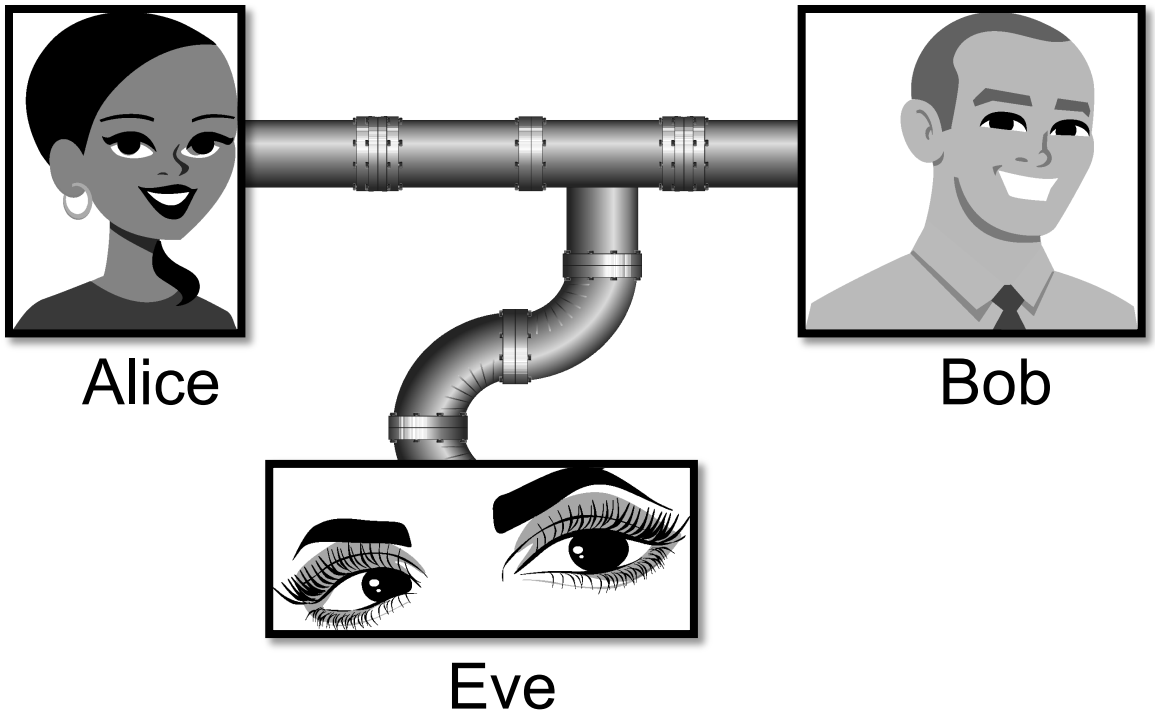
Terminology

- ***Threat***
  - ***possibility of an unauthorized attempt to***
    - access or manipulate information
    - render a system unreliable or unusable
- ***Vulnerability***
  - ***known or suspected flaw in software or design that exposes to***
    - unauthorized disclosure of info
    - system intrusion (ability to control system state)
- ***Attack***
  - ***execution of a plan to carry out a threat by exploiting a vulnerability***
- ***Intrusion***
  - ***successful attack***

Attack and Intrusion often used interchangeably!

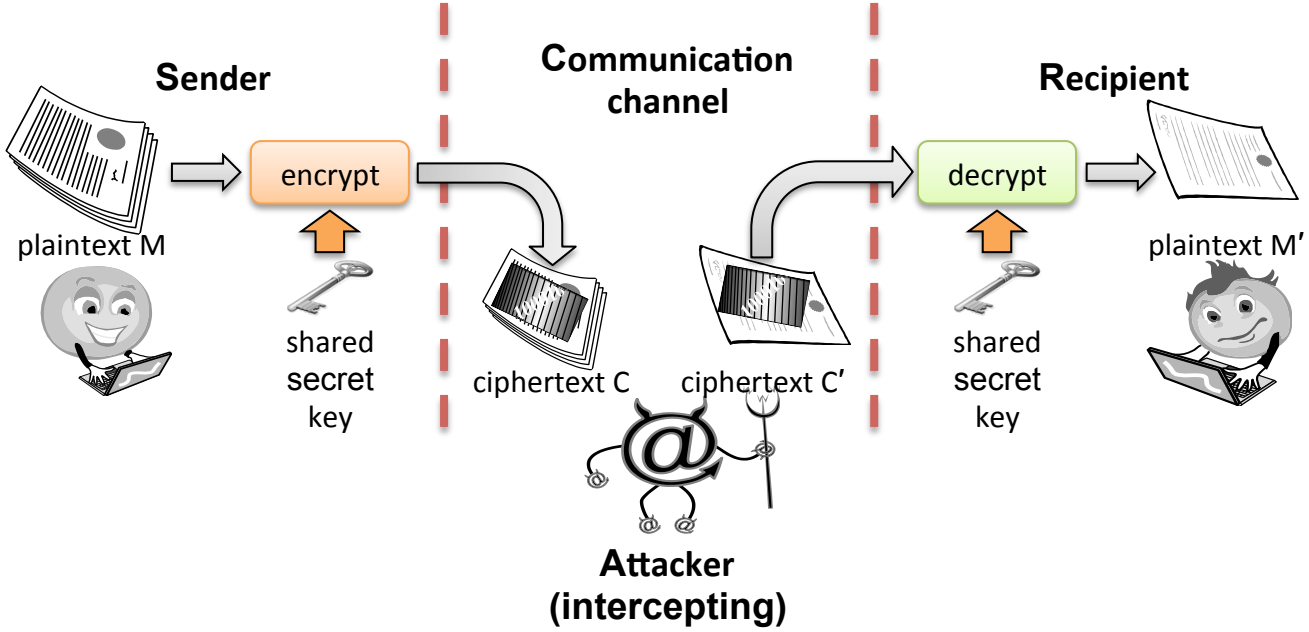
Threats and Attacks

- **Eavesdropping:** the interception of information intended for someone else during its transmission over a communication channel.



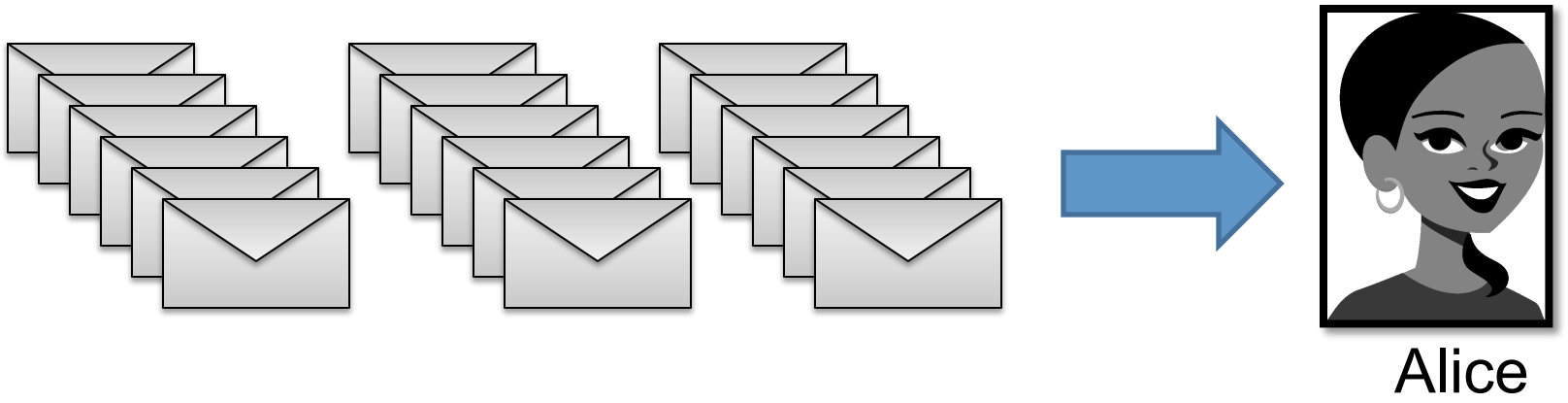
Threats and Attacks

- **Alteration:** unauthorized modification of information.
  - **Example:** the **man-in-the-middle attack,** where a network stream is intercepted, modified, and retransmitted.



Threats and Attacks

- **Denial-of-service:** the interruption or degradation of a data service or information access.
  - **Example:** email **spam,** to the degree that it is meant to simply fill up a mail queue and slow down an email server.



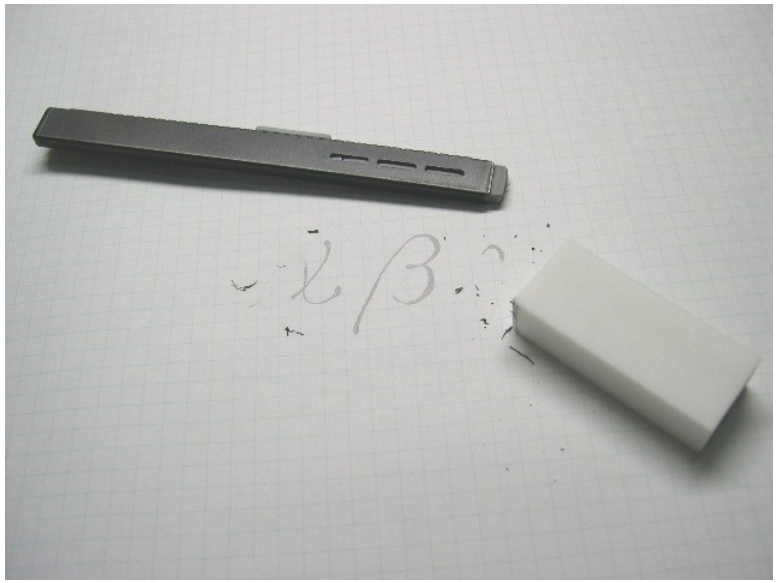
Threats and Attacks

- **Masquerading:** the fabrication of information that is purported to be from someone who is not actually the author.
  - Examples: spoofing, phishing

"From: Alice"
(really is from Eve)

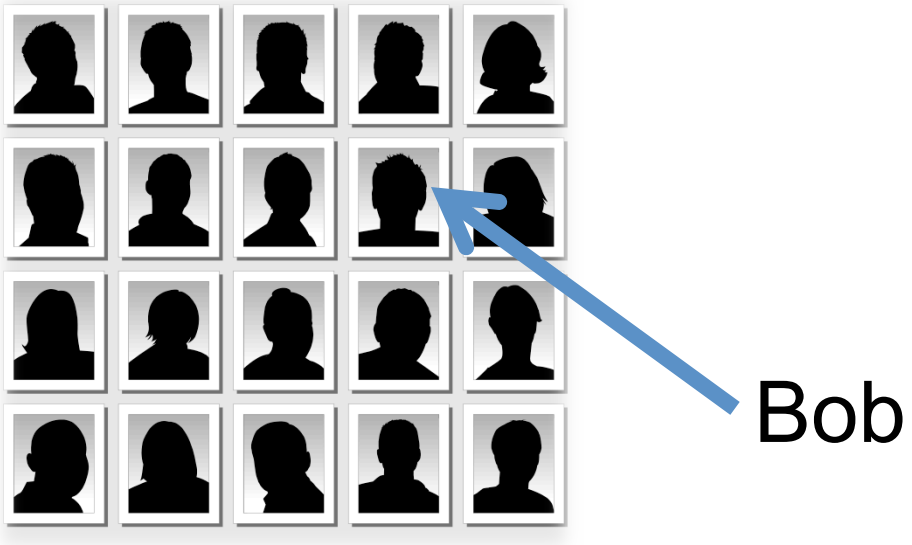
Threats and Attacks

- **Repudiation:** the denial of a commitment or data receipt.
  - This involves an attempt to back out of a contract or a protocol that requires the different parties to provide receipts acknowledging that data has been received.



Threats and Attacks

- **Correlation** and **traceback:** the integration of multiple data sources and information flows to determine the source of a particular data stream or piece of information.
  - Example: traffic watermarking



Social Engineering

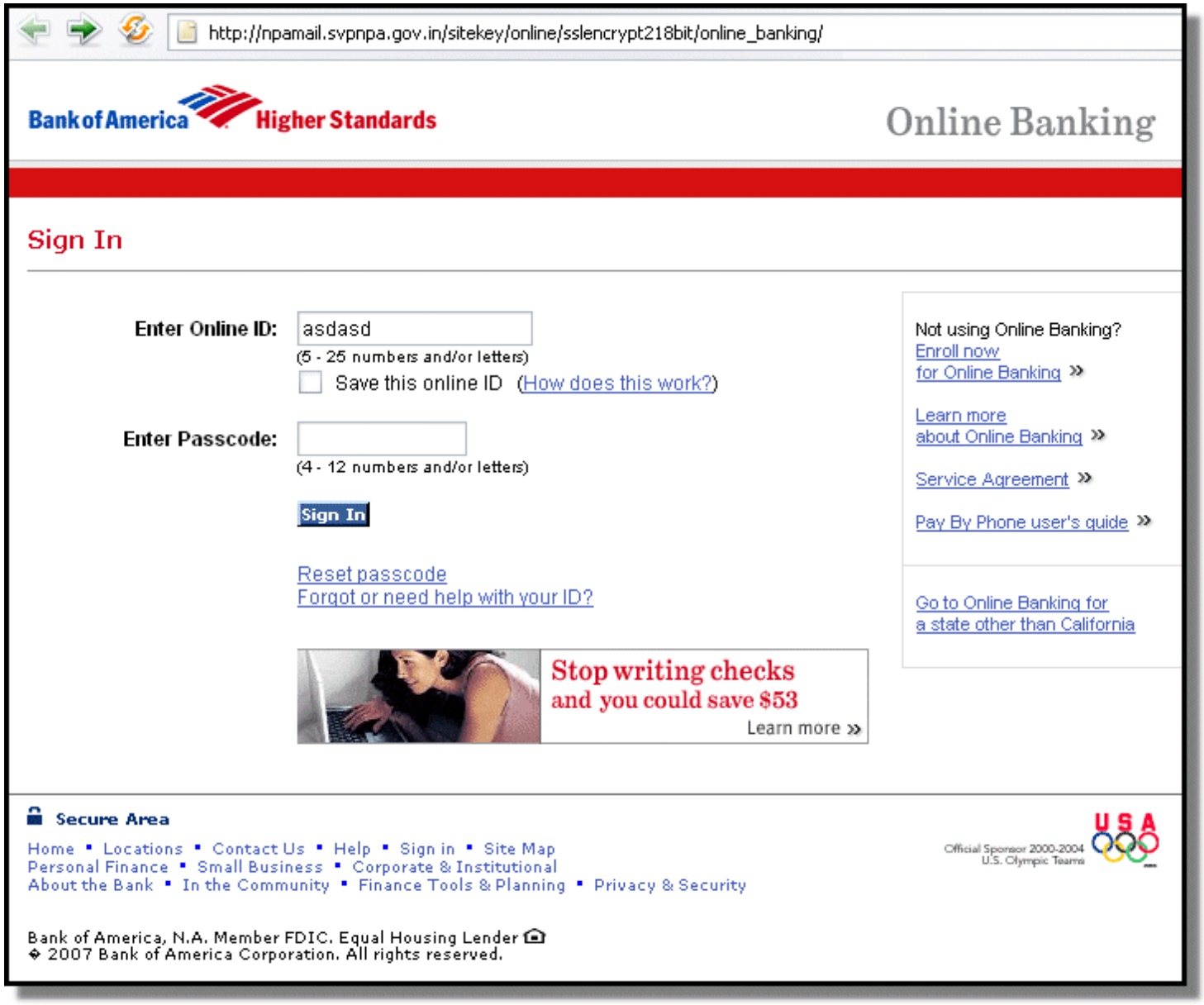
- **Pretexting**: creating a story that convinces an administrator or operator into revealing secret information.
- **Baiting:** offering a kind of "gift" to get a user or agent to perform an insecure action.
- **Quid pro quo:** offering an action or service and then expecting something in return.

Social Engineering

2min 10sec - 12,145 views - 1.7 / 5.00 rating
2916 peoples bookmarked this movie

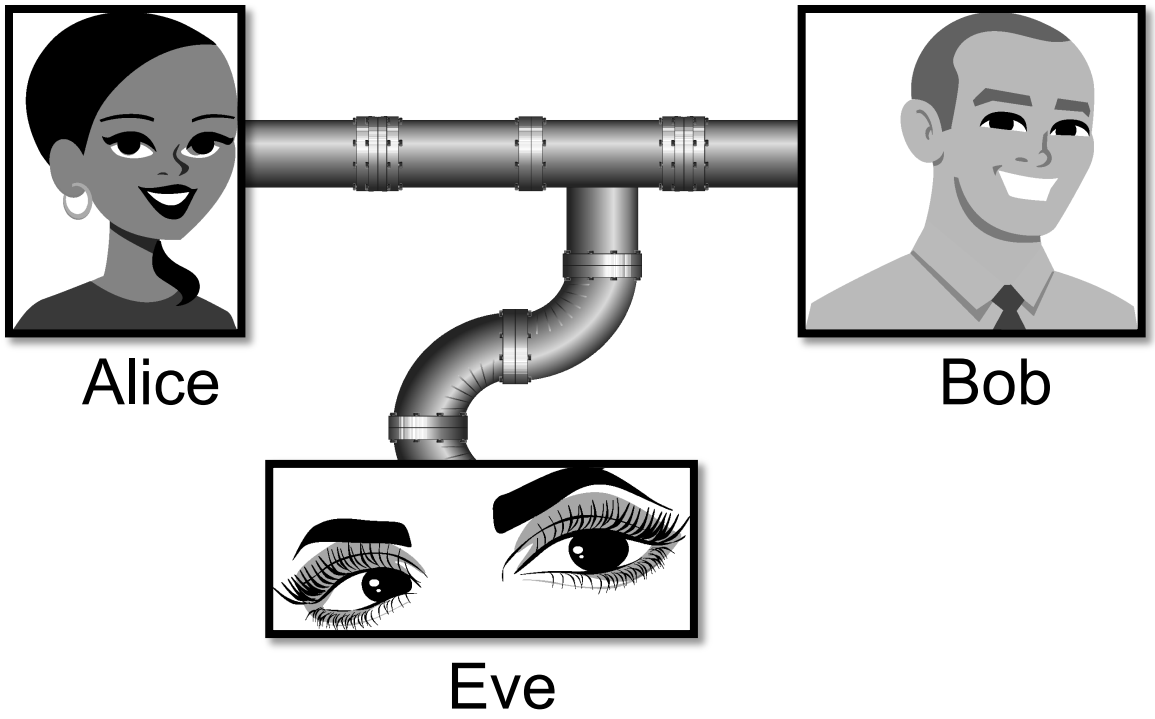
Social Engineering

Social Engineering/Phishing



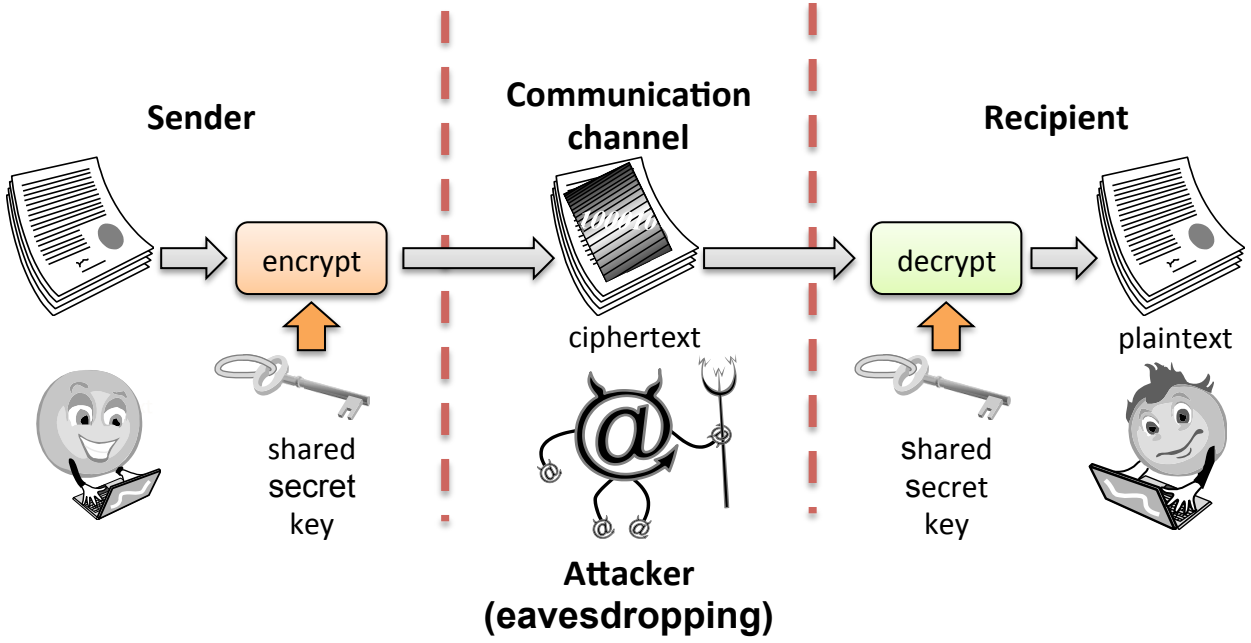
Cryptographic Concepts

- **Encryption**: a means to allow two parties, customarily called Alice and Bob, to establish confidential communication over an insecure channel that is subject to eavesdropping.



Encryption and Decryption

- The message M is called the **plaintext.**
- Alice will convert plaintext M to an encrypted form using an encryption algorithm E that outputs a **ciphertext C for M.**



Encryption and Decryption

- As equations:

$$C = E(M)$$

$$M = D(C)$$

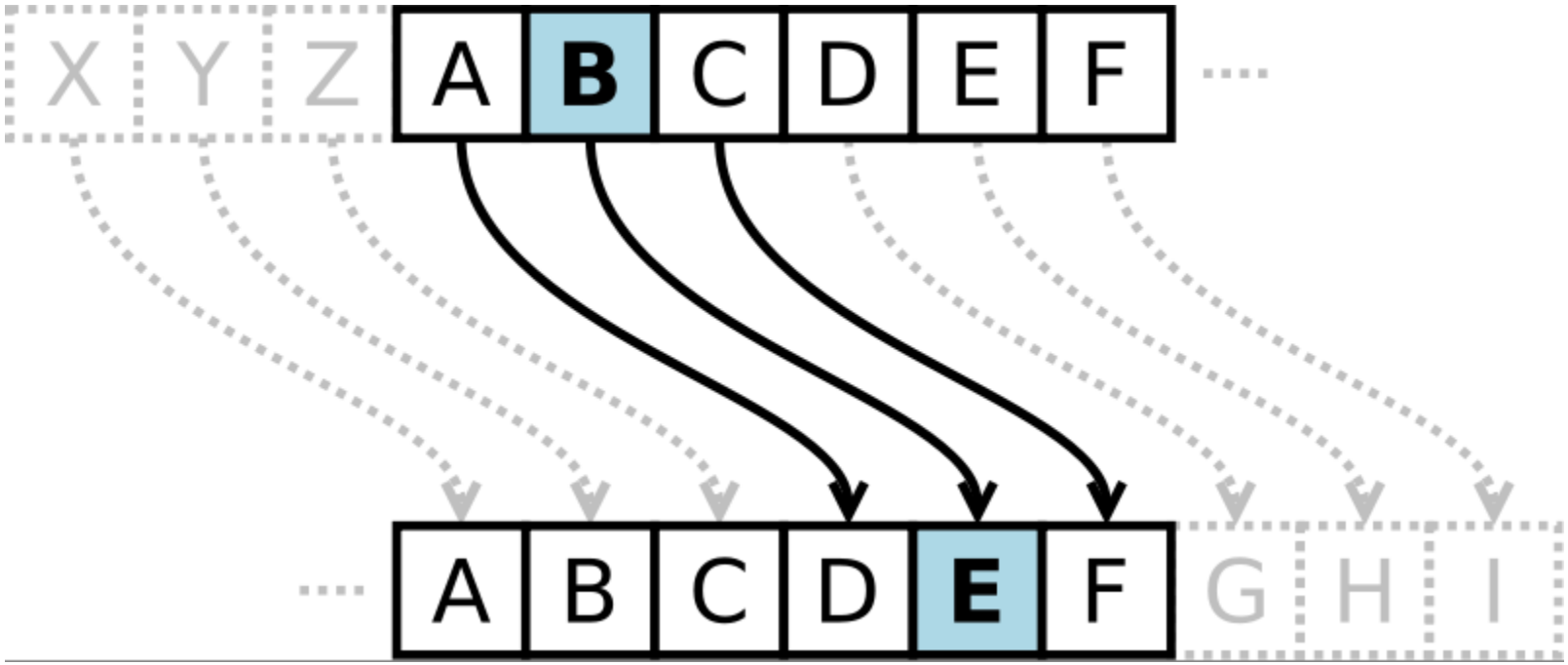
- The encryption and decryption algorithms are chosen so that it is infeasible for someone other than Alice and Bob to determine plaintext M from ciphertext C. Thus, ciphertext C can be transmitted over an insecure channel that can be eavesdropped by an adversary.

Cryptosystem

1. The encryption algorithm to use
2. The decryption algorithm to use
3. The set of encryption keys
4. The set of decryption keys
5. The correspondence between encryption keys and decryption keys
6. The set of possible plaintexts
7. The set of possible ciphertexts

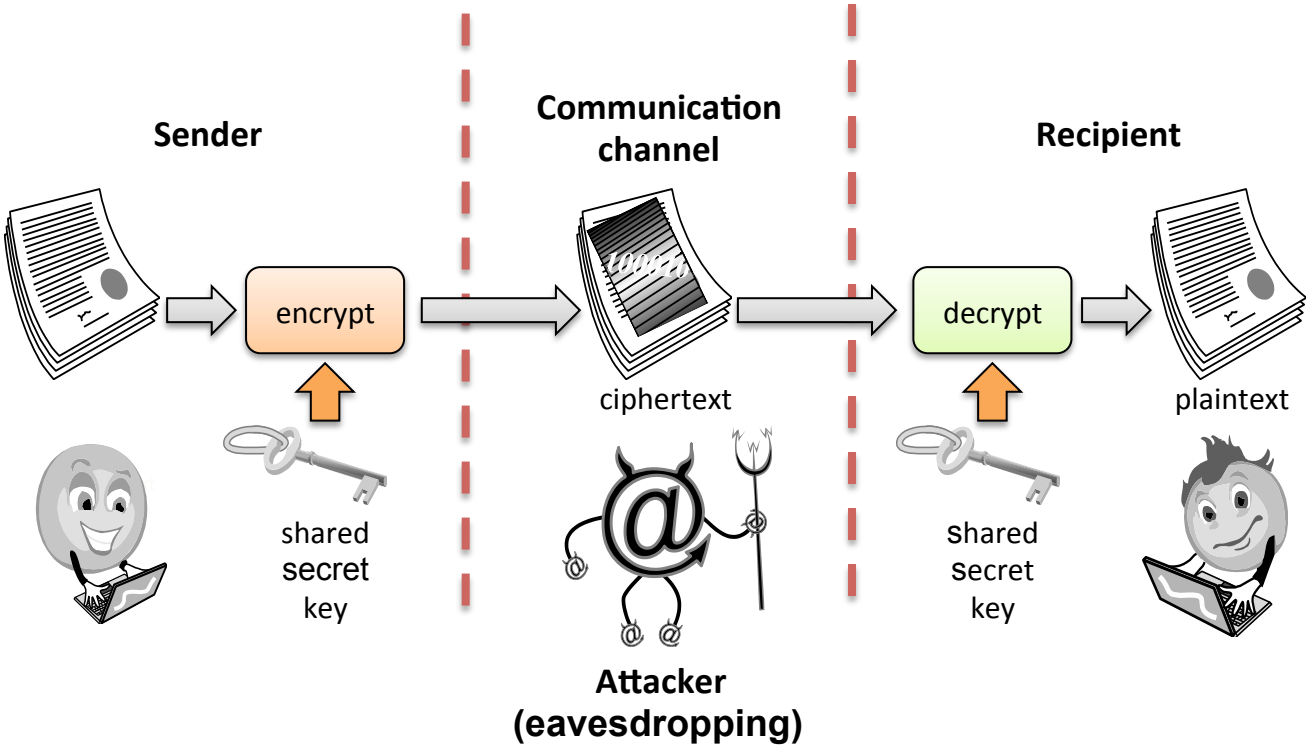
Caesar Cipher

- Replace each letter with the one “three over” in the alphabet.



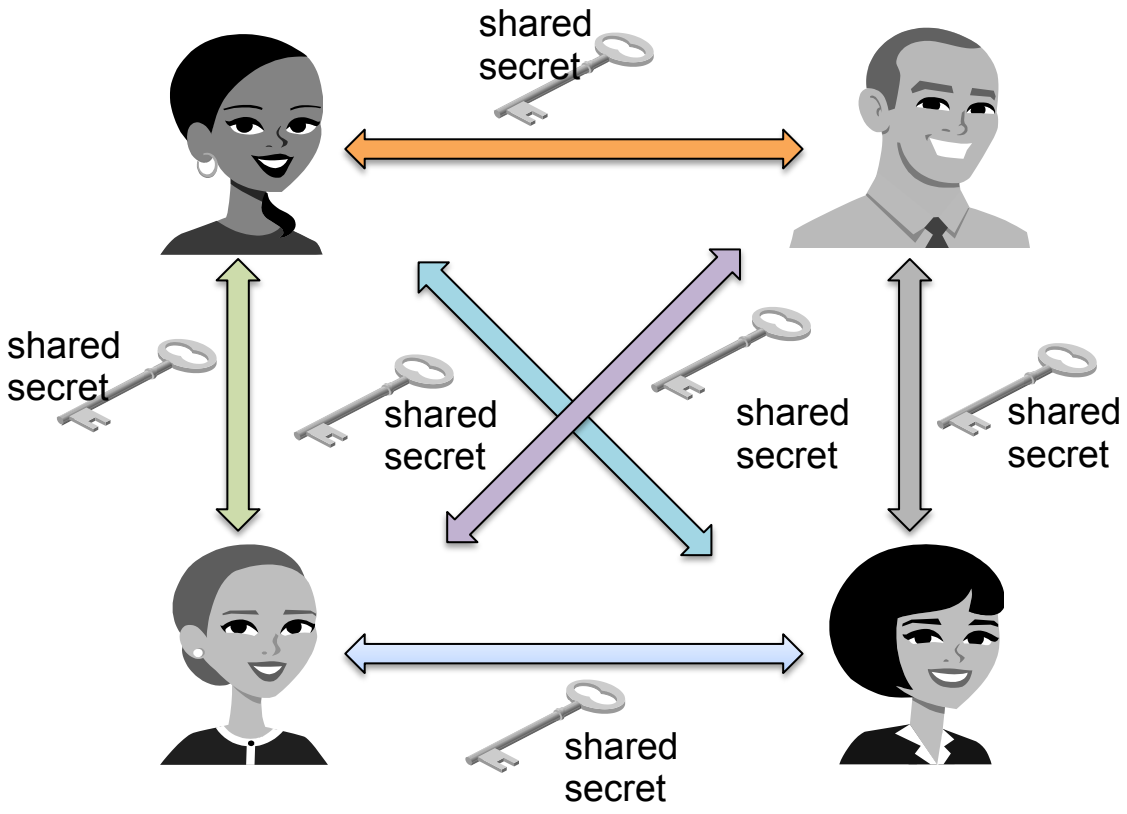
Symmetric Cryptosystems

- Alice and Bob share a secret key, which is used for both encryption and decryption.



Symmetric Key Distribution

- Requires each pair of communicating parties to share a (separate) secret key.



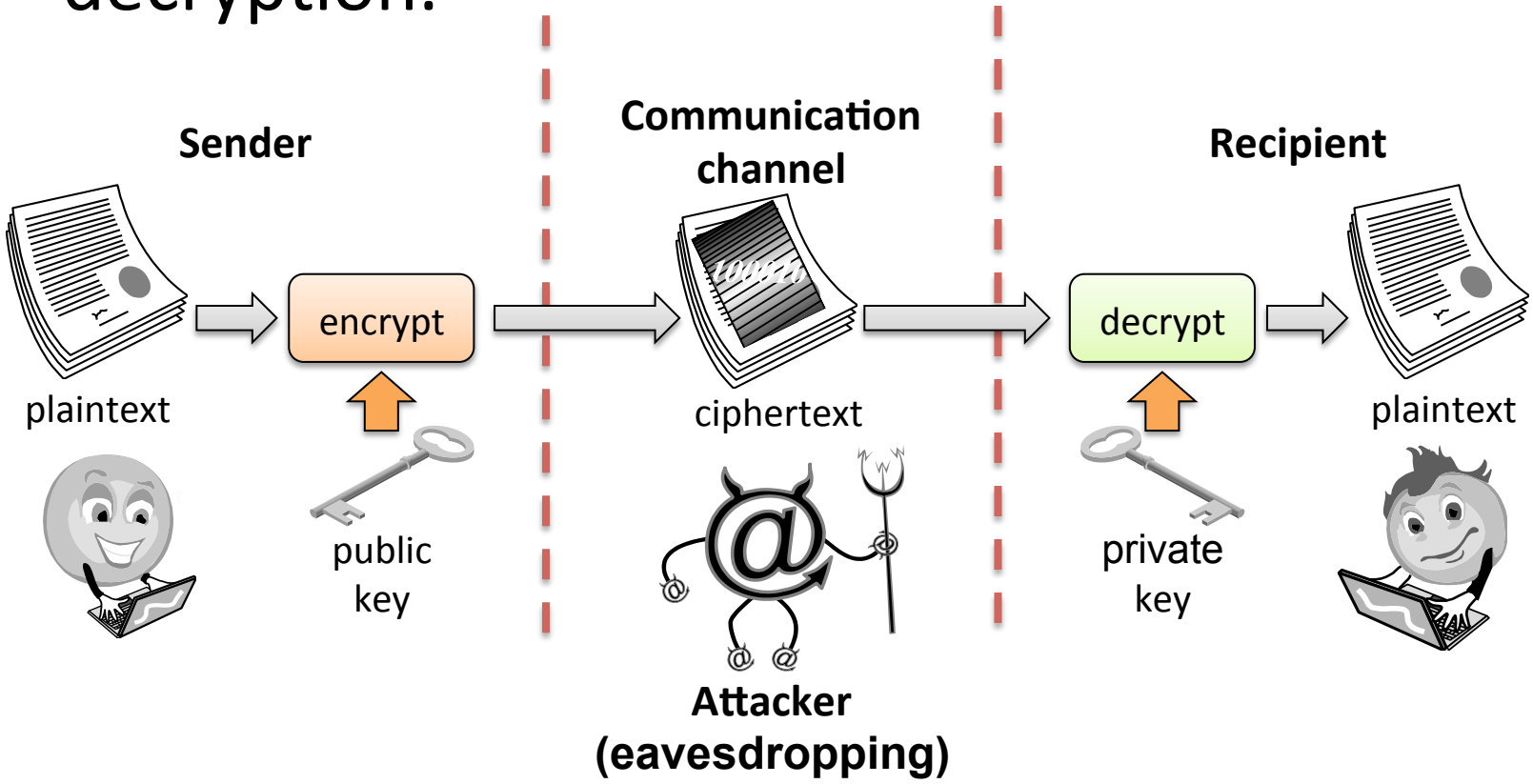
n (n−1)/2 keys

Public-Key Cryptography

- Bob has two keys: a **private key,** SB, which Bob keeps secret, and a **public key,** PB, which Bob broadcasts widely.
  - In order for Alice to send an encrypted message to Bob, she need only obtain his public key, PB, use that to encrypt her message, M, and send the result, $C = E_{PB}(M)$, to Bob. Bob then uses his secret key to decrypt the message as $M = D_{SB}(C)$.

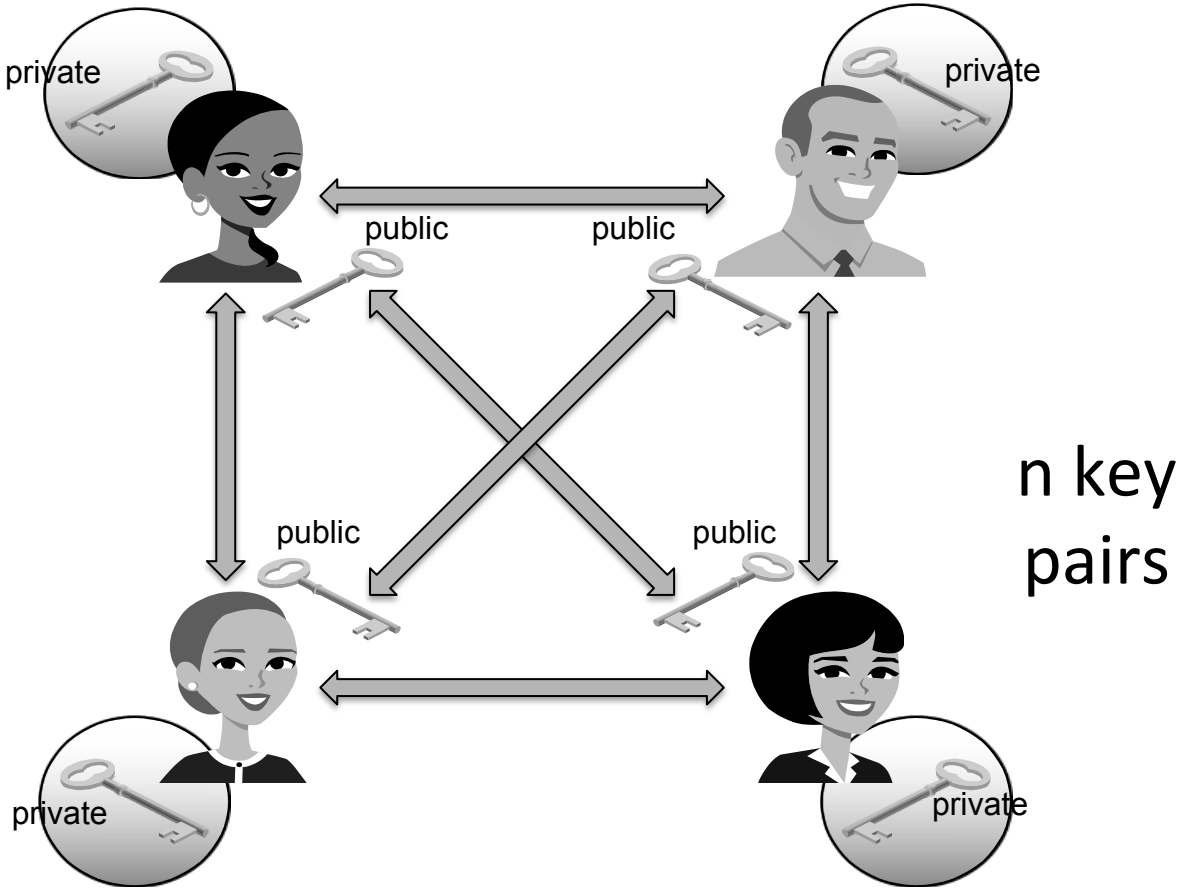
Public-Key Cryptography

- Separate keys are used for encryption and decryption.



Public Key Distribution

- Only one key is needed for each recipient



n key pairs

Digital Signatures

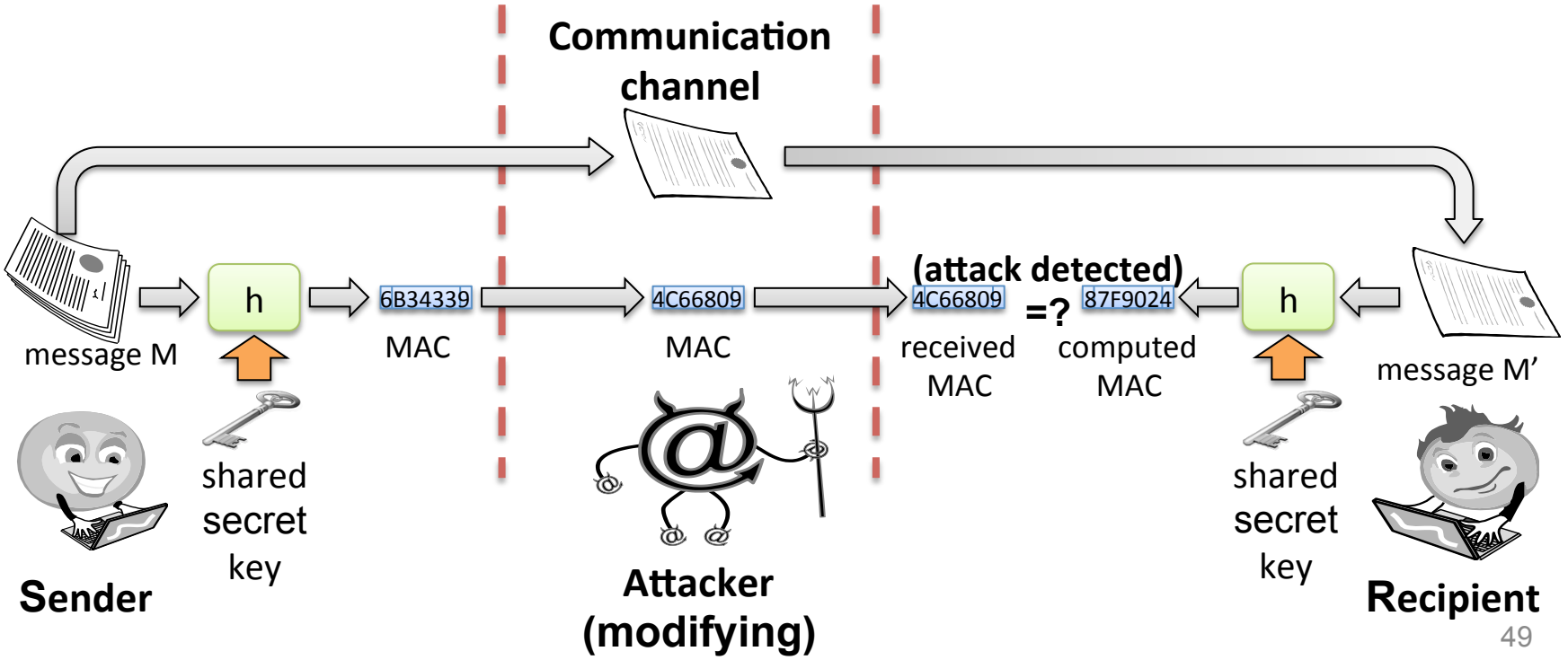
- Public-key encryption provides a method for doing digital signatures
- To sign a message, M, Alice just encrypts it with her private key, SA, creating $C = E_{SA}(M)$.
- Anyone can decrypt this message using Alice's public key, as $M' = D_{PA}(C)$, and compare that to the message M.

Cryptographic Hash Functions

- A checksum on a message, M, that is:
- **One-way**: it should be easy to compute Y=H(M), but hard to find M given only Y
- **Collision-resistant:** it should be hard to find two messages, M and N, such that H(M)=H(N).
- **Examples:** SHA-1, SHA-256.

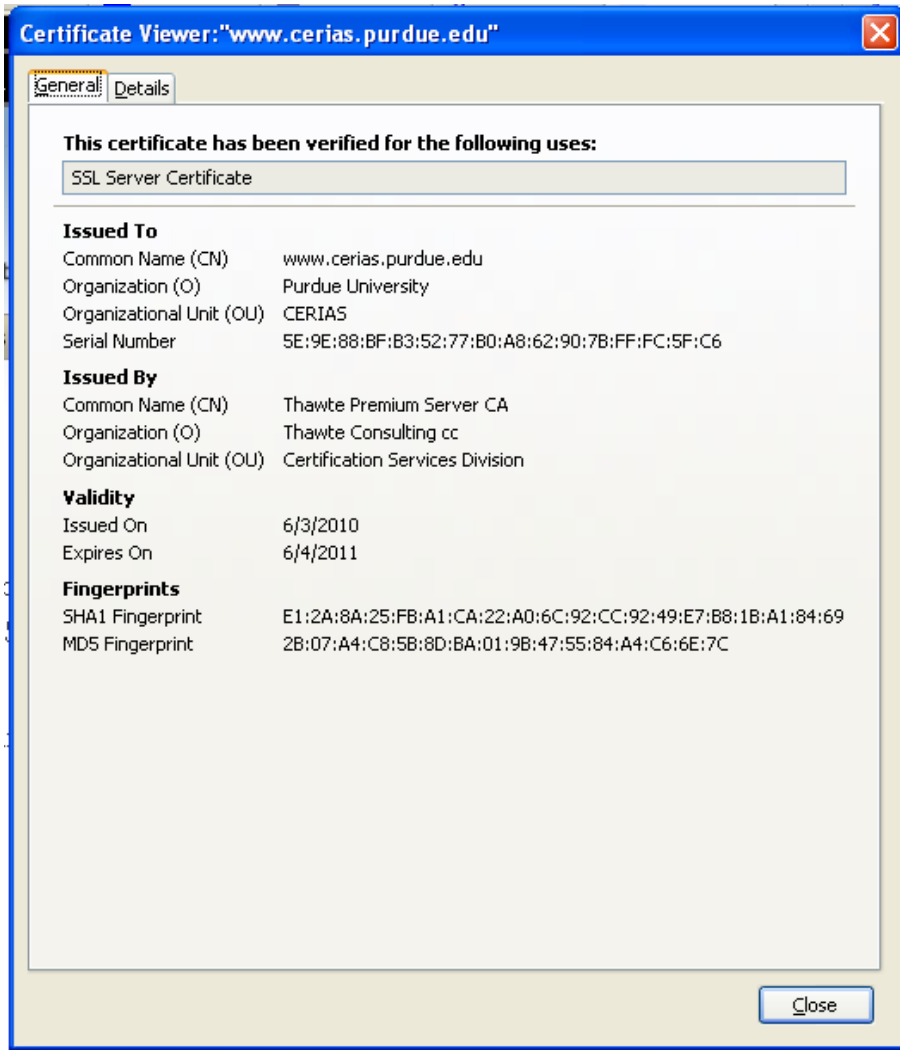
Message Authentication Codes

- Allows for Alice and Bob to have data integrity, if they share a secret key.
- Given a message M, Alice computes H(K||M) and sends M and this hash to Bob.



Digital Certificates

- **certificate authority (CA)** digitally signs a binding between an identity and the public key for that identity.

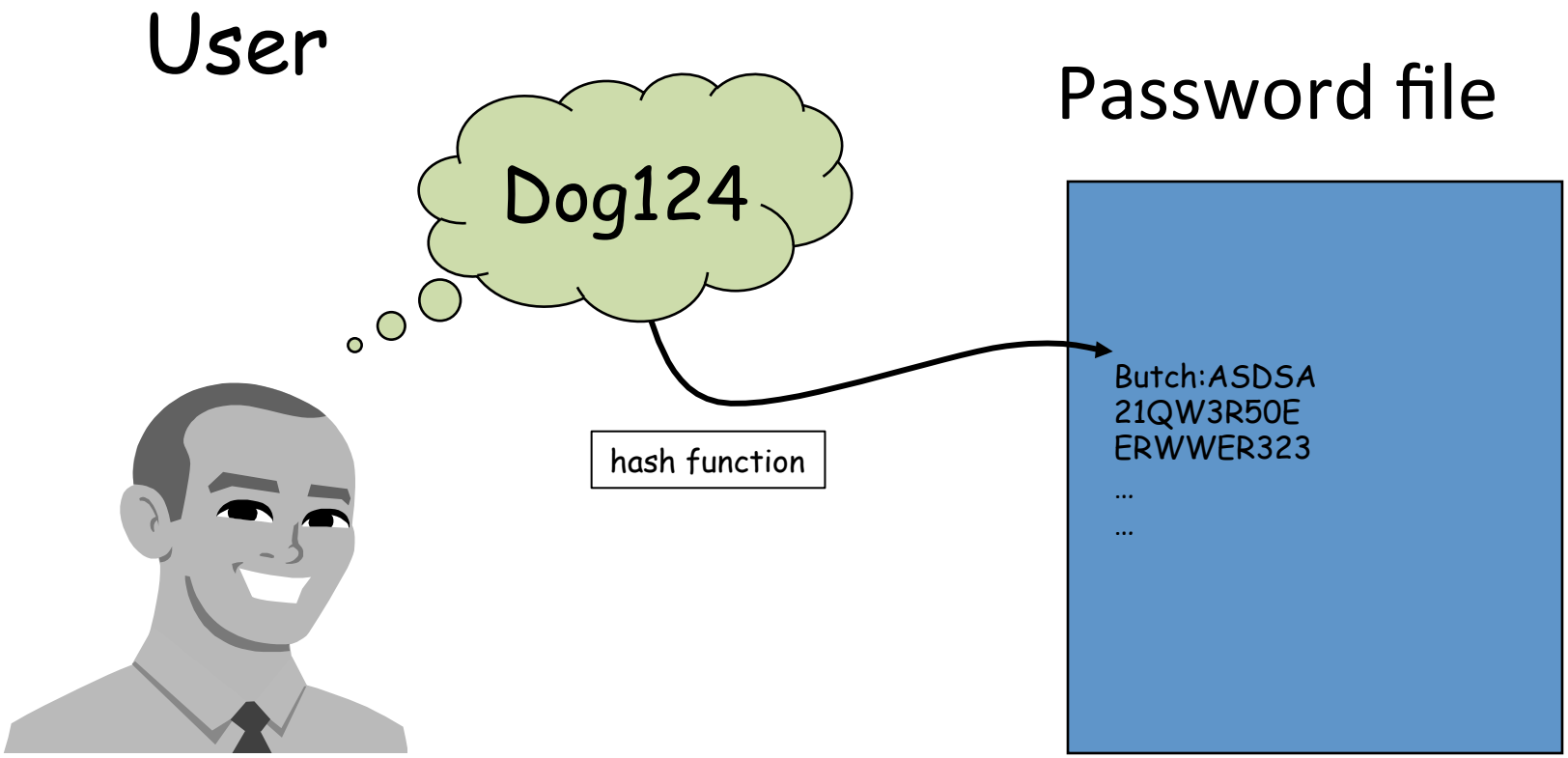


Passwords

- A short sequence of characters used as a means to authenticate someone via a secret that they know.

- Userid: _____
- Password: _____

How a password is stored?



https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet

Strong Passwords

What is a strong password

- UPPER/lower case characters
- Special characters
- Numbers

When is a password strong?

- Seattle1
- M1ke03
- P@$$w0rd
- TD2k5secV

Password Complexity

A fixed 6 symbols password:

- Numbers
  $10^6 = 1{,}000{,}000$
- UPPER or lower case characters
  $26^6 = 308{,}915{,}776$
- UPPER and lower case characters
  $52^6 = 19{,}770{,}609{,}664$
- 32 special characters (&, %, $, £, “, |, ^, §, etc.)
  $32^6 = 1{,}073{,}741{,}824$

94 practical symbols available

- $94^6 = 689{,}869{,}781{,}056$

ASCII standard 7 bit 2^7 =128 symbols

- $128^6 = 4{,}398{,}046{,}511{,}104$

Odd characters make passwords safer

Password Length

26 UPPER/lower case characters = 52 characters
10 numbers
32 special characters
=> 94 characters available

5 characters: 94^5 =	7,339,040,224
6 characters: 94^6 =	689,869,781,056
7 characters: 94^7 =	64,847,759,419,264
8 characters: 94^8 =	6,095,689,385,410,816
9 characters: 94^9 =	572,994,802,228,616,704

Longer passwords are better

Password Validity: Brute Force Test

Password does not change for 60 days

how many passwords should I try for each second?

5 characters:	1,415 PW /sec
6 characters:	133,076 PW /sec
7 characters:	12,509,214 PW /sec
8 characters:	1,175,866,008 PW /sec
9 characters:	110,531,404,750 PW /sec

Secure Passwords

A strong password includes characters from at least three of the following groups:

Group	Example
Lowercase letters	a, b, c, ...
Uppercase letters	A, B, C, ...
Numerals	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Non-alphanumeric (symbols)	( ) ` ~ ! @ # $ % ^ & * - + = | \ { } [ ] : ; " ' < > , . ? /
Unicode characters	€, Γ, ƒ, and λ

Use pass phrases eg. "I re@lly want to buy 11 Dogs!"

Topic: Access Control

- **Access control:** rules and policies that limit access to confidential information to those people and/or systems with a "need to know."
  - This need to know may be determined by identity, such as a person's name or a computer's serial number, or by a role that a person has, such as being a manager or a computer security specialist

Topic: Access Control

- Users and groups
- Authentication
- Passwords
- File protection
- Access control lists

- Which users can read/ write which files?
- Are my files really safe?
- What does it mean to be root?
- What do we really want to control?

Access Control Matrices

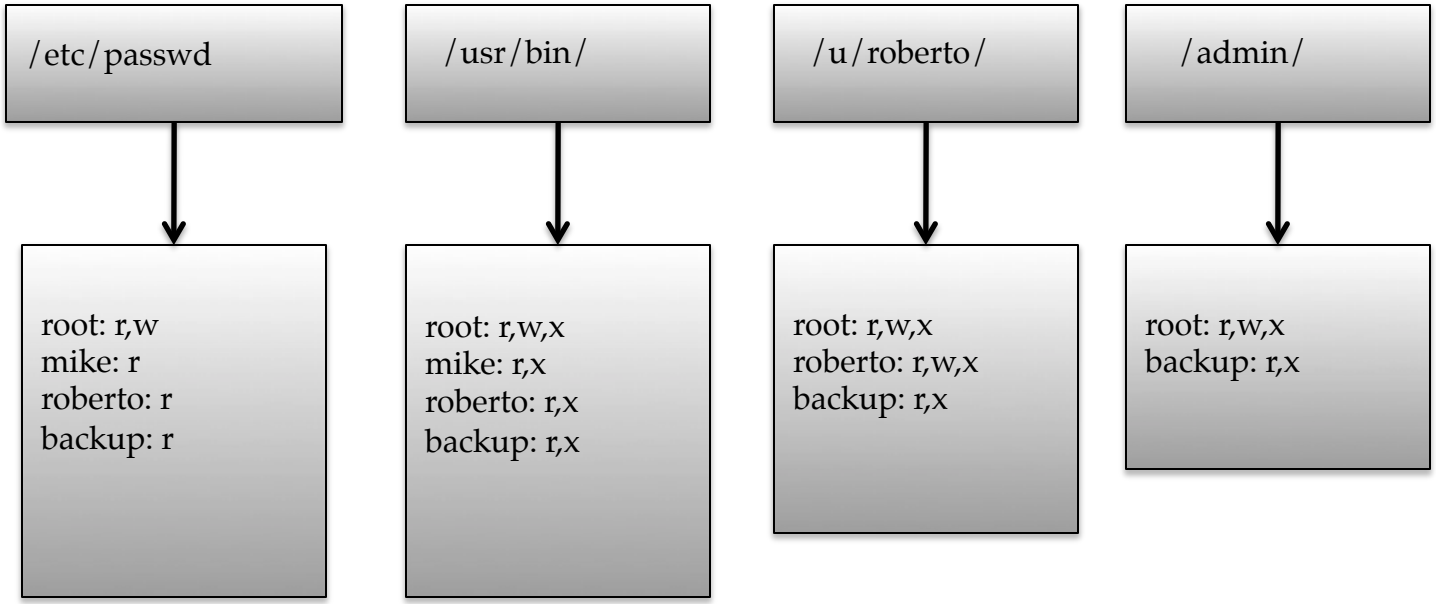
- **A table that defines permissions**
  - Each row of this table is associated with a **subject,** which is a user, group, or system/process that can perform actions.
  - Each column of the table is associated with an **object,** which is a file, directory, document, device, resource, process or any other entity for which we want to define access rights.
  - Each cell of the table is then filled with the access rights for the associated combination of subject and object.
  - Access rights can include actions such as reading, writing, copying, executing, deleting, and annotating.
  - An empty cell means that no access rights are granted.

Example Access Control Matrix

	/etc/passwd	/usr/bin/	/u/roberto/	/admin/
root	read, write	read, write, exec	read, write, exec	read, write, exec
mike	read	read, exec		
roberto	read	read, exec	read, write, exec	
backup	read	read, exec	read, exec	read, exec
. . .	. . .	. . .	. . .	. . .

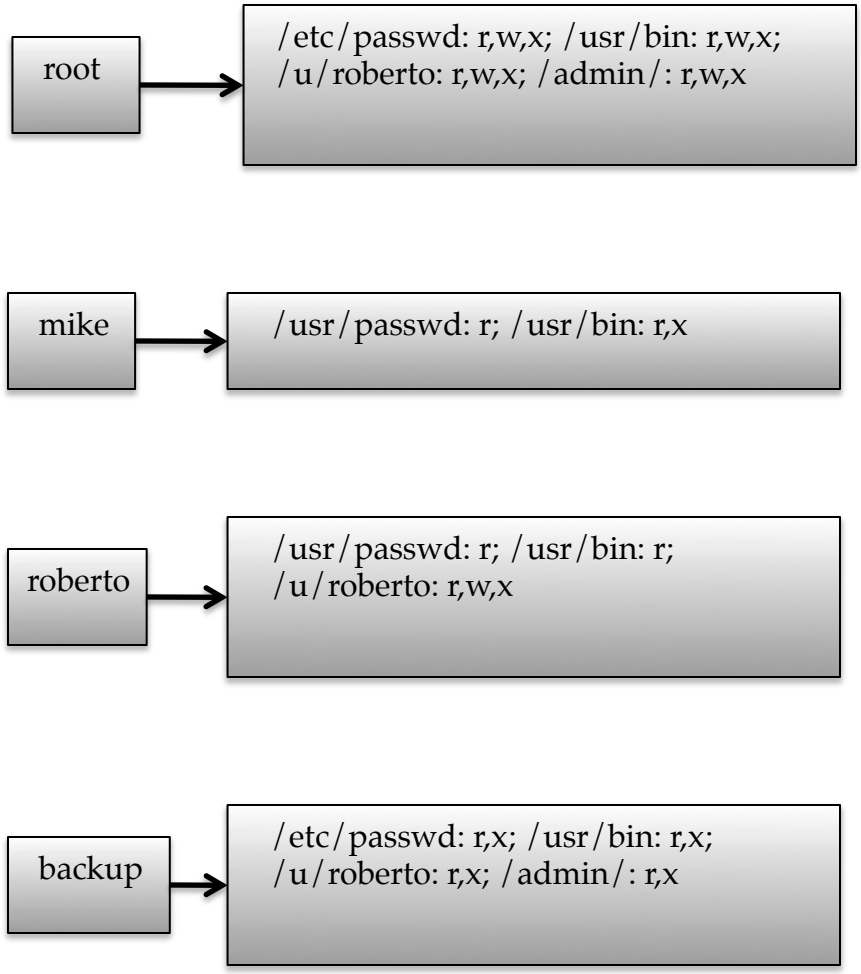
Access Control Lists

- It defines, for each object, o, a list, L, called o's access control list, which enumerates all the subjects that have access rights for o and, for each such subject, s, gives the access rights that s has for object o.



Capabilities

- Takes a subject-centered approach to access control. It defines, for each subject s, the list of the objects for which s has nonempty access control rights, together with the specific rights for each such object.
- Easy for admin to determine what privileges a user/process has

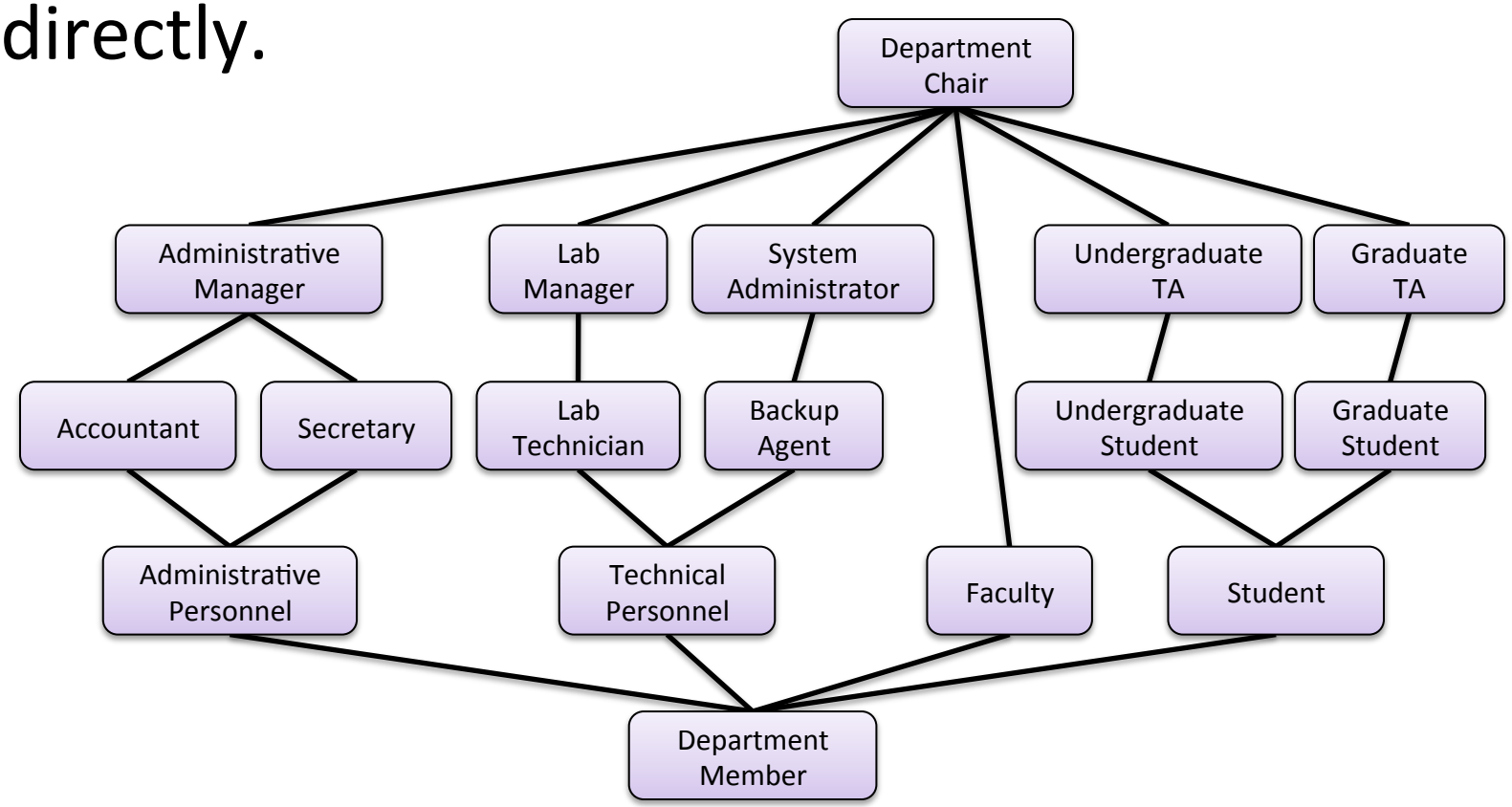


DAC vs. MAC

- Discretionary Access Control (DAC)
  - Users can grant/revoke permissions to access objects they own
- Mandatory Access Control (MAC)
  - Users cannot alter permissions to access any of the objects
  - E.g. only an admin can grant/revoke permissions
  - Implementation example: SELinux

Role-based Access Control

- Define **roles** and then specify access control rights for these roles, rather than for subjects directly.



Roles vs. Groups

- Most things you can do with RBAC can be implemented using groups

- Some differences:
  - Groups:
    - users may have different groups
    - A user automatically inherits all permission of her groups
  - Roles:
    - Users need to consciously invoke a role
      - E.g., log-in as Roberto-admin, or Roberto-faculty
    - Change of role requires different auth credentials
      - E.g., different password for different roles
    - This may prevent some problems due to "role confusion" for a user with multiple roles
      - **Least privilege principle!**
      - E.g., prevent 'rm –rf /' from working when logged-in as 'faculty'
    - Sometimes you want to make sure only one user at a time is in a certain role

The Ten Security Principles

"The protection of information in computer systems" (1975)
http://www.acsac.org/secshelf/papers/protection_information.pdf

- Common-sense principles
- Make systems more secure by design
- Contain damage in case a system is compromised
- Not always correctly implemented

Least privilege

- Each program and user of a computer system should operate with the bare **minimum privileges necessary** to function properly.
  - If this principle is enforced, abuse of privileges is restricted, and the damage caused by the compromise of a particular application or user account is minimized.
  - The military concept of **need-to-know** information is an example of this principle.
  - Example: what can go wrong if you use your system with full admin/root privileges?

Fail-safe defaults

- This principle states that the default configuration of a system should have a ***conservative protection scheme***
- Unless a subject (user or process) is given explicit permission to access an object, ***access should be denied by default***
  - For example, when adding a new user to an operating system, the default group of the user should have minimal access rights to files and services. Unfortunately, operating systems and applications often have default options that favor usability over security.
  - This has been historically the case for a number of popular applications, such as web browsers that allow the execution of code downloaded from the web server.

Economy of mechanism

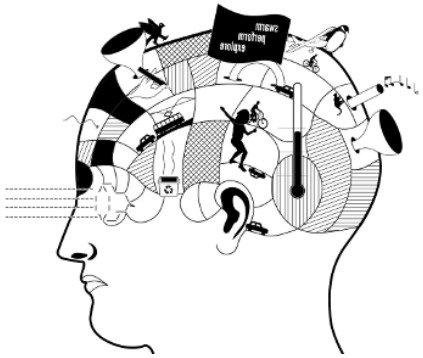
- This principle stresses **simplicity** in the **design** and **implementation** of security measures

- ***Security mechanisms should be as simple as possible***
  - A simple security framework facilitates its understanding by developers and users
  - enables the efficient development and verification (e.g., via code auditing) of enforcement methods

Complete mediation

- The idea behind this principle is that ***every access to a resource must be checked*** for compliance with policies
  - Example: OS always checks file ACLs before granting access to a user/process
  - One should be wary of performance improvement techniques that save the results of previous authorization checks, since permissions can change over time.
  - For example, an online banking web site should require users to sign on again after a certain amount of time, say, 15 minutes, has elapsed.
    - Also, bank may require re-authentication every time a sensitive operation (e.g., money transfer) is requested

Open design



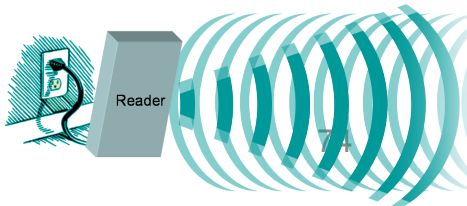
- According to this principle, the security architecture and **design** of a system should be made **publicly available**.
  - For example, for crypto systems, security should rely only on keeping cryptographic keys secret.
  - Open design allows for a system to be scrutinized by multiple parties, which leads to the early discovery and correction of security vulnerabilities caused by design errors
  - The open design principle is the opposite of the approach known as **security by obscurity,** which tries to achieve security by keeping cryptographic algorithms secret and which has been historically used without success by several organizations.

Separation of privilege

- This principle dictates that **multiple conditions** should be required to achieve access to restricted resources or have a program perform some action
  - Example 1: equipment expenditures above $10k may need to be approved by both the CS department and Franklin College
  - Example 2: 'sudo' allows a user to become root only if two conditions are met
    - The user is in the *sudoers* or *admin* group
    - The user enters his/her account password
  - Example 3: two-factor authentication in Gmail
  - Example 4: Nuclear missile launch requires two authorized people to confirm the order

Least common mechanism

- In systems with multiple users, mechanisms allowing resources to be **shared by more than one user should be minimized**.
  - Shared resources provide a channel through which info can flow, and should be minimized
  - Example: side-channel attacks
  - Other example: Attackers can DDoS a website (e.g., amazon) to deprive it from profits from legitimate users. This is possible because the website is shared by attackers and legitimate users. We should restrict the attacker's access to the resource (e.g., through throttling)



Psychological acceptability

- This principle states that user interfaces should be **well designed and intuitive**, and all security-related settings should adhere to what an ordinary user might expect.

- Security mechanism should not make ***the protected resources more difficult to access than if the security mechanisms were not present***
  - Example: ssh access using keys, rather than passwords
  - Higher protection level, same usability
    - Enter key's passphrase, rather than login password
    - Much harder for attacker to gain access to the remote system

Work factor

- According to this principle, the **cost of circumventing** a security mechanism should be compared with the resources of an attacker when designing a security scheme.
  - A system developed to protect student grades in a university database, which may be attacked by snoopers or students trying to change their grades, probably needs less sophisticated security measures than a system built to protect military secrets, which may be attacked by government intelligence organizations.
  - Example: is a 4-digits PIN good enough as a password?
    - 10k combinations may be enough if login can happen only at a physical terminal

Compromise recording

- This principle states that it is desirable to **record the details** of an intrusion even when the intrusion cannot be prevented
  - Internet-connected surveillance cameras are a typical example of an effective compromise record system that can be deployed to protect a building in lieu of reinforcing doors and windows.
  - The servers in an office network may maintain logs for all accesses to files, all emails sent and received, and all web browsing sessions.