# CS370 Operating Systems


## Colorado State University
## Yashwant K Malaiya


## Fall 2025 L27
## Security and Protection

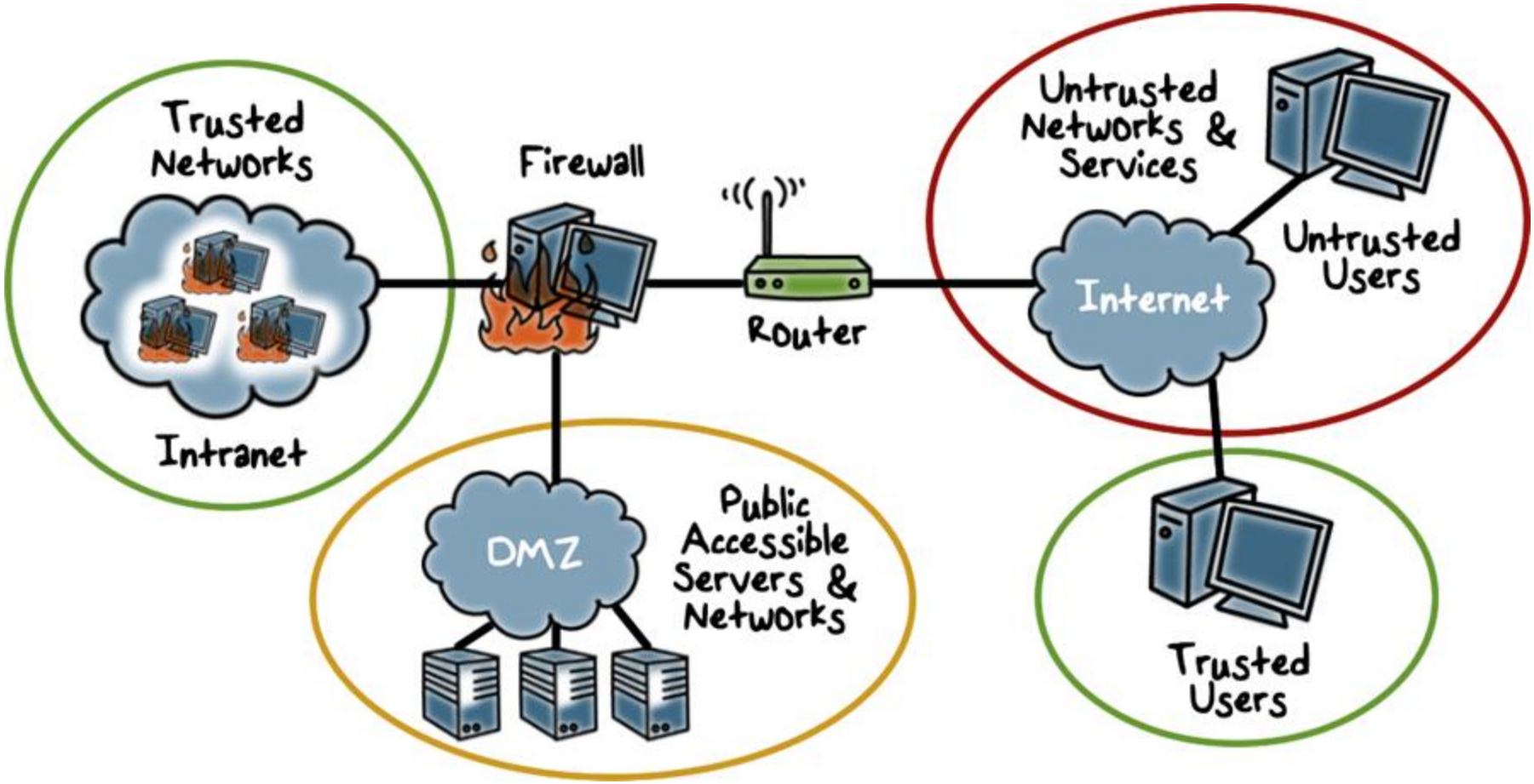
**Slides based on**

- **Text by Silberschatz, Galvin, Gagne**
- **Various sources**

# Security System Architecture

- Networked systems
  - Use of firewalls: Organization wide and system level
  - Address translation
  - Isolation of systems
- Single computing System: OS
  - Multiple levels of priviledges
  - Isolation of
    - processes,
    - cgroups,
    - virtual machines

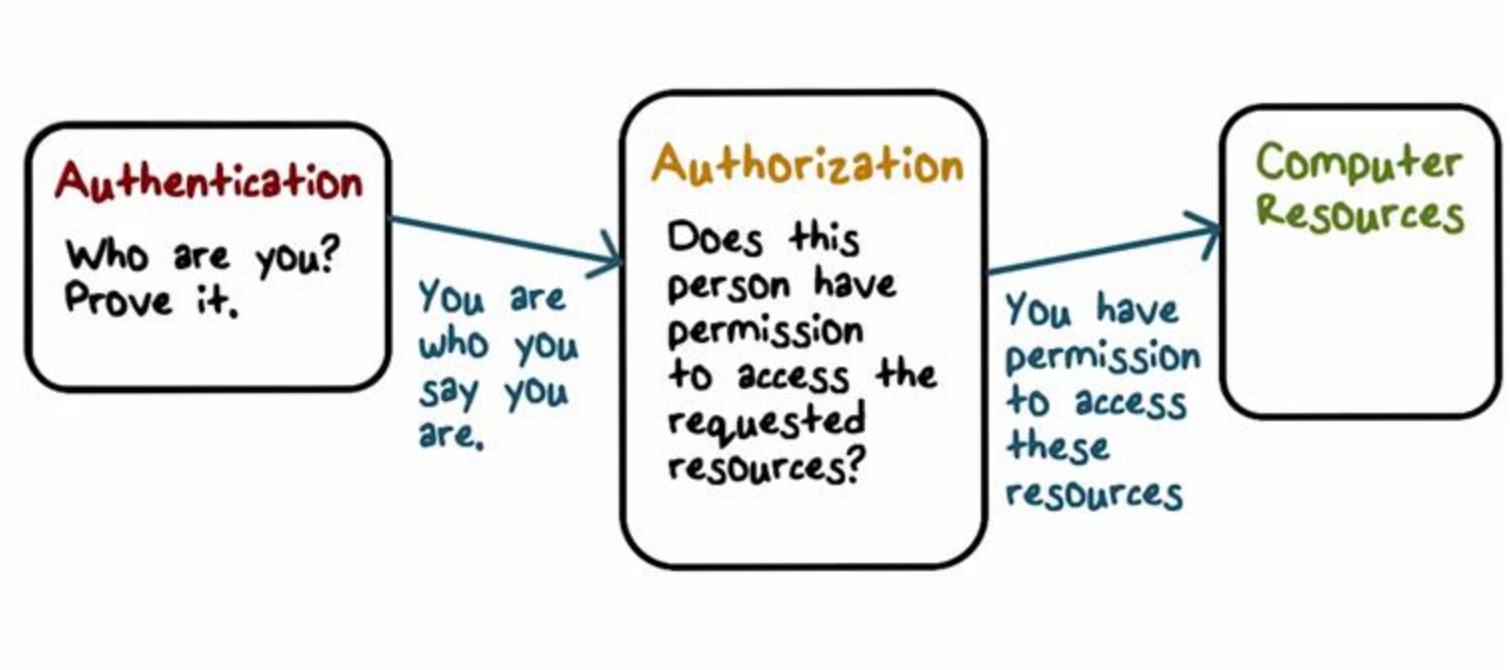
# Firewalls




DMZ: “Demilitarized zone”, distributed firewalls, From Georgia Tech
Note multiple levels of trust.

Colorado State University

# Authentication




Georgia Tech



Colorado State University

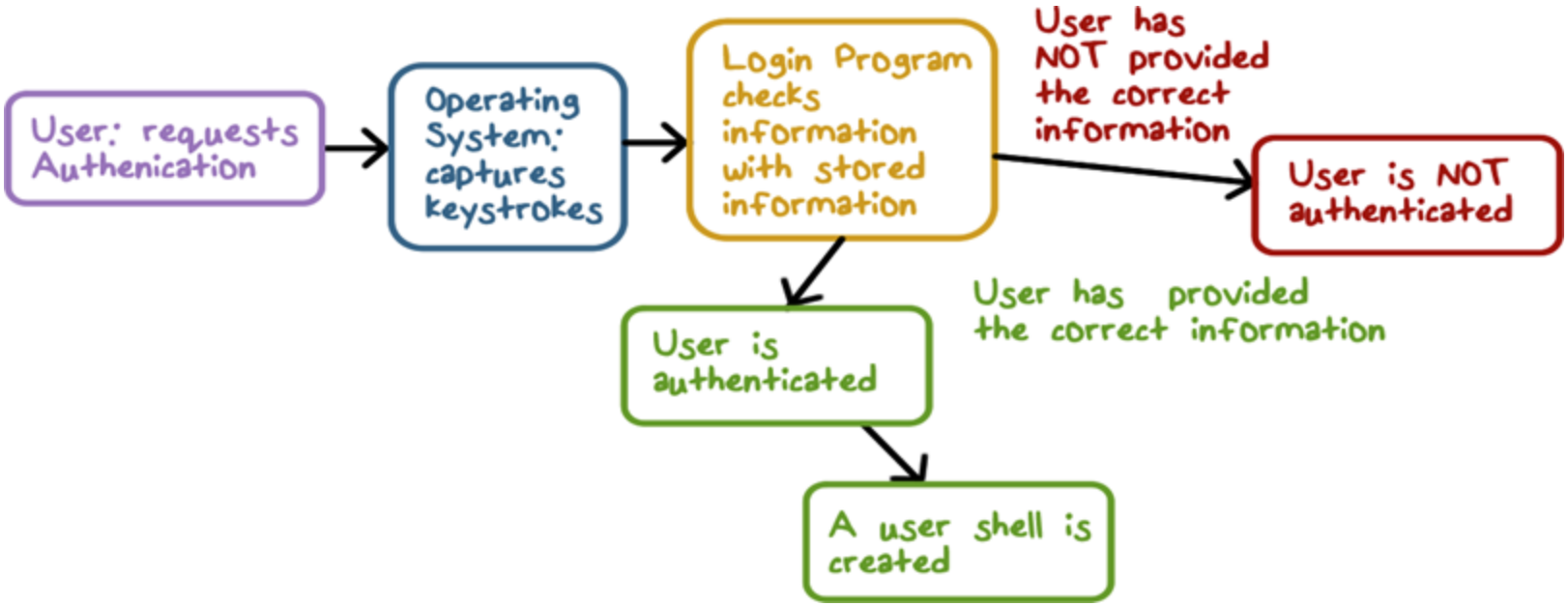
# Authentication Methods

Three existing and two new.

- Something a user knows
  - Password, answers to questions
- Something a user has
  - Ex. Id card, Phone
- Something a user is
  - Biometric (face, iris, fingerprint)
- Somewhere you are geographically
- Something you do based on recognizable patterns of behavior

- Can be multifactor to reduce false positives
- After-access confirmation

# Implementation: Password based Authentication




The system must provide a trusted path from keyboard to the OS.

Georgia Tech



Colorado State University

# Password authentication

Possible approaches

1. Store a list of passwords, one for each user in the system file, readable only by the root/admin account
   - Why the admin need to know the passwords?
   - If security is breached, the passwords are available to an attacker. No longer used.
2. Do not store passwords, but store something that is derived from them
   - Use a hash function and store the result
     - Dictionaries of hashed passwords exist. “Salt” is added to make cracking harder.

# Security Challenges

- Password guessing
  - List of bad passwords. 123456, password, ..
- Brute force guessing
  - more later
- Good passwords are the ones harder to remember
- May be stolen using
  - keyloggers,
  - compromised websites where same password was used
  - eavesdropping *(Alice, Bob and Eve?)*
- Solution: multi-factor authentication

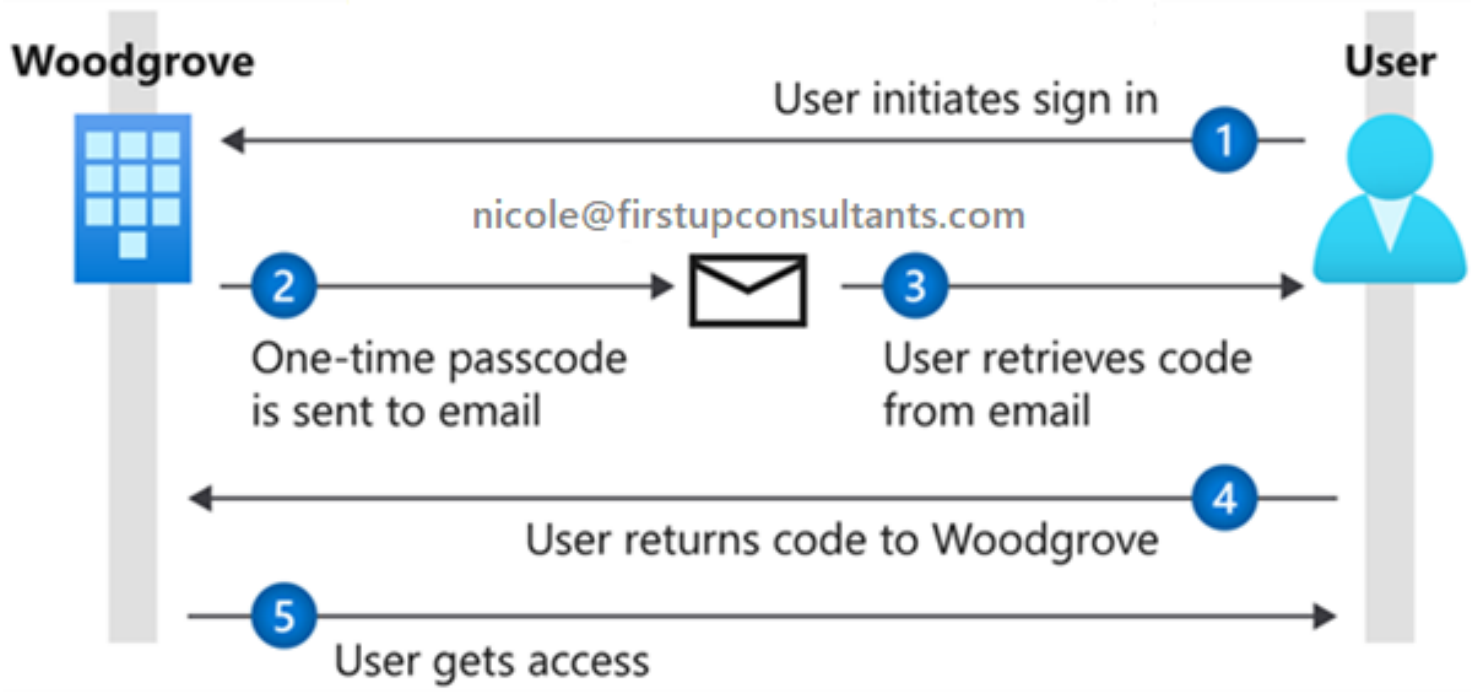
# Biometric Authentication

- Fingerprints (finger swipes)
- Keystroke dynamics
- Voice
- Retina scans

Issues

- Feature value distribution or a range
- False positives and negatives

# Smartphone OTP Authentication




OTP (one-time passcode authentication) texted or email to unique phone number/email address.



Colorado State University

# Project Updates

**Suggested: Slides: max 20. Video: max 15 min.**

**Option A**

- **Project Slides** and videos for both section need to be posted 24 hours in advance of the viewing period.
- Sec 001: Groups 1A -16A: post slides & videos by Sun 12/07/25 2 PM
- Sec 001: 17A - 33A: post slides & videos by Tues 12/09/25 2 PM
- Sec 801: All Groups post slides & videos by Mon 12/08/25 2 PM

**Option B**

- Sec 001: Schedule your 15-min demo (Mon-Wed) with the assigned TA by Thurs 12/04/25, and
  - Post your slides by Tues 12/09/25 2 PM.
  - Post your videos by Tues 12/09/25 2 PM.
- Sec 801: Post your slides and videos by Tues 12/09/25 2 PM.
  - Please follow instructions for videos, including responding to specific questions and being visible in the video. No demos schedules with the TAs.

# Sec 801 online Option B

- No need to schedule demos with TAs.
- Please see Option B: Project Presentation Requirements for 801 Section, including
  - video should not exceed 15 min.
  - at the beginning of the video please do a brief introduction of yourself and your teammates
  - identify each component of the hardware by pointing to them to show that it satisfies the requirements (sensors, actuators, computer it is communicating with, etc.)
- Please address these questions:
  - What was your biggest takeaway from doing this project?
  - biggest hurdle that you encountered while developing the project?
  - Does the project have any aspects that were not initially part of the plan (i.e. you had to add something you didn't expect to make it work)?
  - Are there any aspects of your implementation that you think will stand out compared to similar projects?
  - What two specific attributes of your finished project have you evaluated? How?

# Peer Evaluation

- You must view and evaluate one quarter of the Research and Development projects from both sections.
  - Evaluate Novelty/Interest, Technical/Research, Presentation.
  - Use scores 10, 9, 8, 7: a quarter (7) each
  - 33 groups in Sec 001, 21 in sec 801, 1/4 means 14
- In addition, you need to critically review two assigned project reports and provide feedback to the authors.
- Evaluation for and details will be shared soon.

# Quantitative Security

**Colorado State University**
**Yashwant K Malaiya**

**Risk and its components**

**CSU Cybersecurity Center**
**Computer Science Dept**

# Risk: Formal definition

Definition: The Risk due to an adverse event $e_i$ is

$$Risk_i = Likelihood_i \times Impact_i$$

- $Likelihood_i$: Probability of the adverse event i occurring within a specific time-frame.
  - The time-frame is often chosen to be a year. Note that the probability of an adverse event happening depends on the duration of the time-frame.
  - Probability is a number between 0 and 1.
- $Impact_i$: The impact of the adverse event, measured in monetary terms.
  - Note that impact me be direct or indirect.
  - Common units are dollars (US$)#.

# US$ is a common and convenient scale. Non-monetary losses, including human life, can be converted into US$, if you are a business or insurance company .

# Risk: Possible Actions

How to handle risk?

Example: Credit card fraud

- Risk acceptance
  - Ex: fraud cost paid through fees charged to merchants
- Risk mitigation
  - Ex: install anti-fraud technology, adds to costs
- Risk avoidance
  - downgrade high-risk cardholders to debit or require additional verification: lost time/business
- Risk transfer
  - buy cyber-insurance to cover excess losses

# Risk as a composite measure

- Likelihood can be split in two factors

  $\text{Likelihood}_i = P\{\text{A security hole}_i \text{ is exploited}\}.$

  $= P\{\text{hole}_i \text{ present}\}.$

  $P\{\text{exploitation} \mid \text{hole}_i \text{ present}\}$

- $P\{\text{hole}_i \text{ present}\}$: an internal attribute of the system.
- $P\{\text{exploitation} \mid \text{hole}_i \text{ present}\}$: depends on circumstances outside the system, including the adversary capabilities and motivation.
- In the literature, the terminology can be inconsistent.

Caution: In classical risk literature, the internal component of Likelihood is termed "Vulnerability" and external "Threat". Both are probabilities. There the term "vulnerability" does not mean a security bug, as in computer security.

# Annual value of the countermeasure

Cost/benefit analysis of countermeasures

- A countermeasure reduces the ALE by reducing one of its factors.

COUNTERMEASURE_VALUE

= (ALE_PREVIOUS – ALE_NOW) – COUNTERMEASURE_COST

Where ALE_PREVIOUS: ALE before implementing the countermeasure.

ALE_NOW: ALE after implementing the countermeasure

COUTERMEASURE_COST: *annualized* cost of countermeasure

- The COUNTERMEASURE_VALUE should be positive.

# ROI

- Businesses often use a term Return on Investment (ROI. it can be defined as

  ROI =

  COUNTERMEASURE_VALUE/COUNTERMEASURE_COST

- Given the following, what is the ROI?
  - ALE_PREVIOUS = $50,000
  - ALE_NOW=$25,000
  - COUNTERMEASURE_COST = $10,000
  - ROI = (50,000-25,000-10,000)/10,000 = 1.5

  (This assumes an annual perspective)

# Likelihood & Impact scales

- Quantitative or descriptive levels
  - Number of levels may depend on resolution achievable
- Scale: Logarithmic, Linear or combined
  - A logarithmic scale is natural when the numbers involved vary by several orders of magnitude.
- Risk = Likelihood x Impact
  - May be rewritten as

    Log(Risk) = Log(Likelihood) + Log( Impact)
- If the term "Score" is proportional to Log value
  - Risk score = Likelihood score + Impact score
  - Adding scores valid if scores represent logarithmic values.
  - Example:
    - Likelihood = 10%, impact = $100,000 ⇒ **Risk = $10,000**
    - Scores: Log(0.10) = -1, log (100000) = 5 ⇒ **Risk score = 4**

$10^2 = 100$
$\text{Log}_{10}\ 100 = 2$
Log implies base 10, if not specified

# Risk Matrix

- Likelihood and Impact divided into levels
  - Each level quantitatively/qualitatively defined
- Cells marked by the overall risk
  - Low, Medium, High, Extreme etc.
- *Equal risk regions* along the diagonal, valid provided score scales are logarithmic.

| Likelihood | Consequences | | | | |
|---|---|---|---|---|---|
| | Insignificant | Minor | Moderate | Major | Severe |
| Almost certain | M | H | H | E | E |
| Likely | M | M | H | H | E |
| Possible | L | M | M | H | E |
| Unlikely | L | M | M | M | H |
| Rare | L | L | M | M | H |



Colorado State University

21

# Risk Matrix $Risk_i = Likelihood_i \times Impact_i$

| **LIKELIHOOD** (probability) How likely is the event to occur at some time in the *(Linear Scale time specific matrix)* | **CONSEQUENCES** What is the Severity of injuries /potential damages / financial impacts (if the risk event actually occurs)? *(Logarithmic Scale, property industry specific matrix)* | | | | |
|---|---|---|---|---|---|
| | Insignificant | Minor | Moderate | Major | Catastrophic |
| | No Injuries First Aid<br>No Envir Damage<br><< $1,000 Damage | Some First Aid required<br>Low Envir Damage<br><< $10,000 Damage | External Medical<br>Medium Envir Damage<br><<$100,000 Damage | Extensive injuries<br>High Envir Damage<br><<$1,000,000 Damage | Death or Major Injuries<br>Toxic Envir Damage<br>>>$1,000,000 Damage |
| Almost certain - expected in normal circumstances (100%) | MODERATE RISK | HIGH RISK | HIGH RISK | CRITICAL RISK | CRITICAL RISK |
| Likely – probably occur in most circumstances (10%) | MODERATE RISK | MODERATE RISK | HIGH RISK | HIGH RISK | CRITICAL RISK |
| Possible – might occur at some time. (1%) | LOW RISK | MODERATE RISK | HIGH RISK | HIGH RISK | CRITICAL RISK |
| Unlikely – could occur at some future time (0.1%) | LOW RISK | MODERATE RISK | MODERATE RISK | HIGH RISK | HIGH RISK |
| Rare - Only in exceptional circumstances 0.01%) | LOW RISK | LOW RISK | MODERATE RISK | MODERATE RISK | HIGH RISK |

Example:

Likely x Moderate
= (10/100) x $100,000
= $10,000 High

Colorado State University

# Quantitative Security

**Colorado State University**
**Yashwant K Malaiya**

**Encryption Architectures**

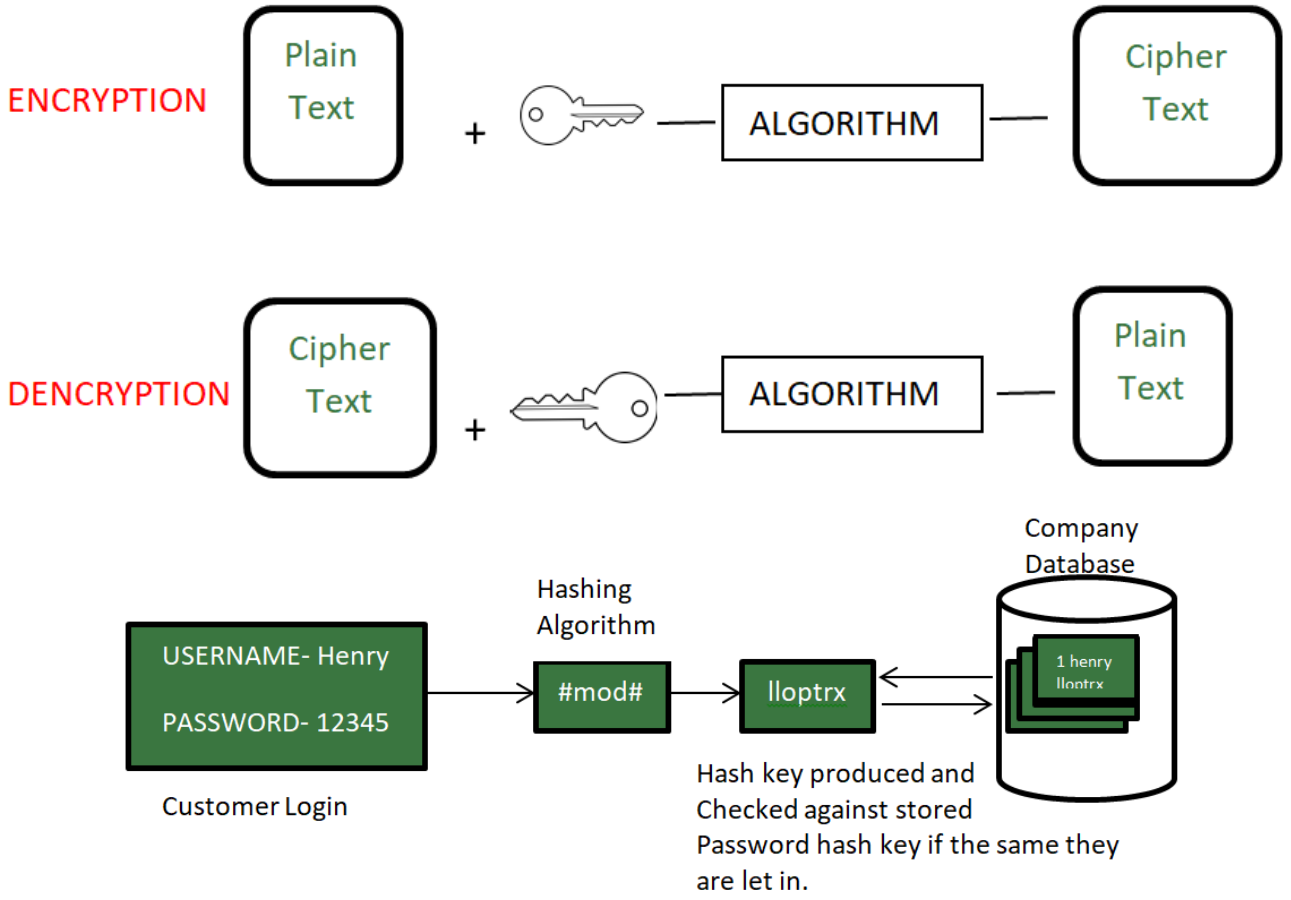
**CSU Cybersecurity Center**
**Computer Science Dept**

# Encryption

Human readable information is *PlainText*. It can be encrypted into *cyphertext*.

- **Hashing**: one way only. Decryption is not possible.
- Symmetric: The same key is used for encryption and decryption using specific mathematical algorithms. Key size 40-2028 bits.
- Asymmetric: A pair of related keys is generated: Public key and Private key.
  - PlainText encrypted using one of them can only be decrypted by the other.
  - Private key is kept private. It cannot be generated using the Public key.

# Encryption vs Hashing




- Impossible to reconstruct password from hash but ..

# Hashed Passwords

- Key idea: store encrypted versions of passwords
  - Use one-way cryptographic hash functions
  - Examples: md5, sha1, sha256, sha512 etc (128, 160, 256, 512 bits)
- Cryptographic hash function transform input data into scrambled output data
  - Deterministic: hash(A) = hash(A)
  - High entropy: small change, significant effect
    - md5('security') = e91e6348157868de9dd8b25c81aebfb9
    - md5('security1') = 8632c375e9eba096df51844a5a43ae93
    - md5('Security') = 2fae32629d4ef4fc6341f1751b405e45
  - Collision resistant
    - Locating A' such that hash(A) = hash(A') takes a long time
    - Example: $2^{21}$ tries for md5

Ack: Northeastern U

# Hashed Password Example

User: cbw

md5('p4ssw0rd') =
2a9d119df47ff993b662a8ef36f9ea20

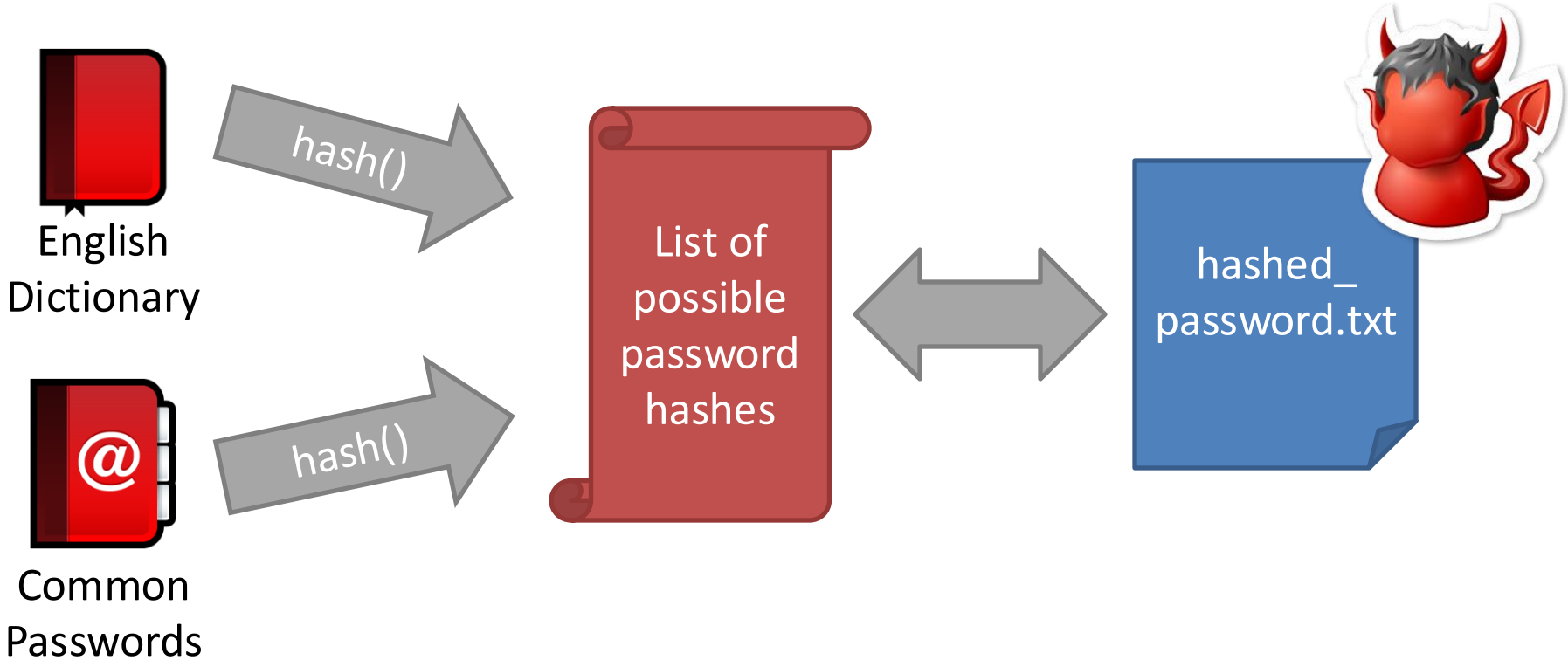
md5('2a9d119df47ff993b662a8ef36f9ea20')
= b35596ed3f0d5134739292faa04f7ca3

| hashed_passw | xt |
|---|---|
| cbw | 2a9d119df47ff993b662a8ef36f9ea20 |
| sandi | 23eb06699da16a3ee5003e5f4636e79f |
| amislove | 98bd0ebb3c3ec3fbe21269a8d840127c |
| bob | e91e6348157868de9dd8b25c81aebfb9 |

# Attacking Password Hashes

- Problem: users choose poor passwords
  - Most common passwords: 123456, password
  - Username: cbw, Password: cbw
  - Common password patterns: 111111, qwerty, ..
- Default passwords (*password*, *default*, *admin*, *guest etc*) if not changed can be a security hazard.
- Weak passwords enable dictionary attacks using Rainbow tables
  - Pre-computed tables of common passwords, dictionary entries etc and their hashes
  - Hundreds of gigabytes or terabytes of data

# Dictionary Attacks




- Common for 60-70% of hashed passwords to be cracked in <24 hours [Free Password Hash Cracker]

# Pwned Passwords

- Pwned Passwords are **501,636,842** real world passwords previously exposed in data breaches. They're searchable online:
- https://haveibeenpwned.com/Passwords
- The searches are anonymized using a mathematical property called *k*-anonymity

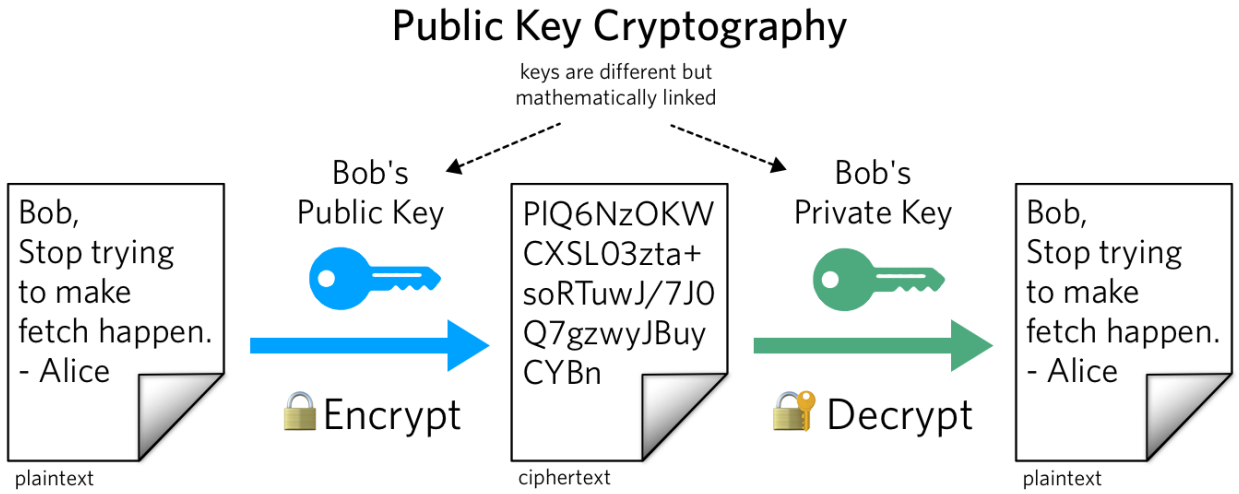
# Hardening Password Hashes

- Key problem: cryptographic hashes are deterministic
  - hash('p4ssw0rd') = hash('p4ssw0rd')
  - This enables attackers to build lists of hashes
- Solution: make each password hash unique
  - Add a **salt** to each password before hashing
  - hash(salt + password) = password hash
  - Each user has a *unique, random* salt
  - Salts can be stores in plain text
- Online tools: dcode.fr others

# Public Key Cryptography?

In public-key cryptography, also known as asymmetric cryptography, each entity has two keys:

- Public Key — to be shared
- Private Key — to be kept secret
- These keys are generated at the same time using an algorithm and are mathematically linked. When using the RSA algorithm, the keys are used together in one of the following ways:






https://www.twilio.com/en-us/blog/developers/tutorials/building-blocks/what-is-public-key-cryptography#What-Is-Public-Key-Cryptography

# Using the Public and the Private key

When using the RSA algorithm, the keys are used together in one of the following ways:

1. Encrypting with a public key

   Use: sending messages only the intended recipient can read.

   Bob encrypts a plaintext message with Alice's public key, then Alice decrypts the ciphertext message with her private key. Since Alice is the only one with access to the private key, the encrypted message cannot be read by anyone besides Alice.

2. Signing with your private key

   Use: verifying that you're the one who sent a message.

   Alice encrypts a plaintext message with her private key, then sends the ciphertext to Bob. Bob decrypts the ciphertext with Alice's public key. Since the public key can only be used to decrypt messages signed with Alice's private key, we can trust that Alice was the author of the original message.

- These methods can also be combined to both encrypt and sign a message with two different key pairs.

# Uses of Public-Key Cryptography

**Uses of Public Key Cryptography**

- SSH
- TLS (HTTPS, formerly SSL)
- Bitcoin
- PGP and GPG
- Authentication
- Public Key Infrastructure (PKI)

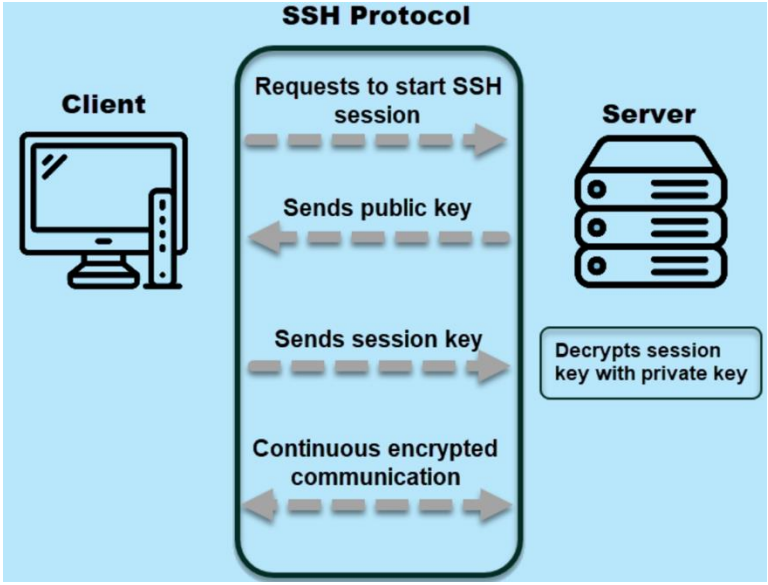
**Public Key Cryptography Algorithms**

- RSA (Rivest–Shamir–Adleman): The current NIST recommendations for RSA is to use at least 2048 bits for your key
- Diffie–Hellman: The Diffie–Hellman approach involves two people building a shared secret key together.
- Elliptic-Curve Cryptography (ECC): smaller key sizes. Using in Bitcoin.

**Note:** Public Key Cryptography is computationally expensive. It is often used to establish a connection and exchange a symmetric key.

# SSH (Secure Shell Protocol) & HTTPS

- When a client initiates a connection to a server and sends a request to start an SSH session, the server responds with the public key. The client uses the public key to verify the server's identity and generates a session key (symmetric).
- Next, the client sends the key to the server in an encrypted form that only the server can decrypt. The server decrypts the code with the private key and uses it to encrypt all future communication with the client.
- Both the client and server use the session key to encrypt all communication, providing confidentiality and integrity. Once the session is complete, both the client and server exchange a final message and close the connection.
- HTTPS works the same way.



# Passkeys

- Passkeys are a standardized form of passwordless authentication that use a combination of
  - a browser API
  - and an authentication device like your phone or computer
- to offer secure, site specific credentials.
- For example, if you generate a passkey to log into acmeinc.com, the passkey only works on acmeinc.com, which helps prevent phishing attacks and account takeovers.
- Passkeys also have built-in multi-factor authentication in the form of device possession and a user gesture like a fingerprint scan or PIN.
- Passkey synchronization with passkey managers like Apple or Google allows users to sign into websites on multiple devices or to restore access if they lose an authenticator.

https://www.twilio.com/en-us/blog/developers/best-practices/passkeys-101

# Passkeys

- Passkeys also have built-in multi-factor authentication in the form of device possession and a user gesture like a fingerprint scan or PIN.
- Passkey synchronization with passkey managers like Apple or Google allows users to sign into websites on multiple devices or to restore access if they lose an authenticator.

# How do passkeys work?

Passkeys use public key cryptography to generate an authentication key pair (a public key and a private key) also known as a credential.

- Public keys are stored on a backend server (either the *relying party* server or their authentication provider's server)
- Private keys are stored on the device where they were generated and in the passkey manager like the iCloud Keychain or Google's passkey manager.
- There are two process for using passkeys.
  - The **registration process** typically occurs after a user has signed into a website or application using an existing method. Key pair is generated and Public key is transmitted.
  - **Authentication Process** When the user returns to sign in later, the process is streamlined and passwordless.
  - Both involve several exchanges.

# Reflecting on Part 1

- System structure and program compilation/execution
- Processes & Threads:
  - creation
  - scheduling
  - termination
- Inter-process communication
  - Synchronization
  - Deadlocks (included in Part 2)

# Part 2

We will review these on next Thursday.

- Virtual and physical address spaces
  - Pages and frames
    - Translation using page tables and TLBs
    - Effective access time
  - Virtual memory
    - Demand paging, page replacement algorithms
  - File systems
    - Disk organization, block allocation, scheduling
    - RAIDs
  - Virtual machines and containers
  - Data centers and cloud
  - Security, access control, authentication, risk