D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Subject Name: Information System

Unit No:04 Unit Name: 4: Systems Design

Faculty: Mrs. Bhavana Alte
Mr. Prathmesh Gunjgur

Unit No: 4 Unit Name: Systems Design

Representing Identity

What is Identity?

- In computing, **identity is a unique representation of an entity** (user, process, device, or system). It is **used to assign privileges, enforce security policies, and enable communication**.
- **Key Components of Identity in Computer Systems:**

1. Principal:
•The **entity being identified** (e.g., a person, device, or software process).
•Example: A logged-in **user account** or a running **system process**.

2. Identifier:
•A **unique value assigned to a principal** (e.g., username, UID, IP address).
•Example: User alice123 or IP 192.168.1.10.

What is Identity?

3. Authentication:

•The **process of verifying an identity** (e.g., passwords, biometrics, certificates).

•Example: **Logging in to Gmail** with a username and password.

4. Authorization:

•Determines **what actions an authenticated identity can perform**.

•Example: **Admin vs Guest accounts** on a website

- **Example:**
- A **Windows user logs in** with a **username and password**.
- The system **authenticates their identity** and grants them **appropriate permissions**.

Authentication vs. Authorization

Concept	Definition	Example
Authentication	Proves who you are (verifying identity).	Logging into Gmail with a password.
Authorization	Determines what actions you are allowed to perform.	A normal user cannot install software, but an admin can.

Representation of Identity

- **Why Identity is Important?**
- **Accountability:** Ensures users and systems **can be traced** (logging, auditing).
- **Access Control:** Determines **who can access what** resources.
- 💡 **Example:**
- In **UNIX systems**, a user has a **User ID (UID)**.
- Every **process started by the user inherits their UID** and corresponding **permissions**.

Challenges in Identity Representation:

- If identities are **not properly managed**, unauthorized users **may gain access**.
- Impersonation or **spoofing** can compromise security.

Files and Object Identity

How Files Are Identified in Systems?
•**Local Naming:** File names, directory paths.
•**Kernel-Level Identification:** Inode numbers, file descriptors
•.
💡 **Example:**
•**UNIX File Naming:**
  •/home/user/document.txt (Path-based identity)
  •**Inode** and **File Descriptors** are used internally by the OS.
🔹 **URLs as Object Identifiers:**
•http://abccorp.com/pub/README
•The file can be constructed dynamically, forwarded, or even invalidated.

- **Security Concerns:**
- **Race Conditions** (two processes accessing the same file simultaneously).
- **File Spoofing Attacks** (malicious redirection of file access).

User Identity in Systems

Different Ways Users Are Identified:

- **Username**: Human-readable identifier.
- **User ID (UID)**: System-assigned unique number.
- **Effective UID**: Used for determining access rights.
- **SetUID Programs:** Temporary privilege escalation.

💡 **Example:**

- **UNIX Login Process:**
    - A user logs in as john_doe, internally stored as UID 1001.
    - When executing certain commands (like sudo), **effective UID changes to 0 (root)**.

🛠 **Security Risks:**

- **SetUID Exploits**: Attackers can misuse high-privilege programs.
- **Privilege Escalation**: Malicious users gaining unauthorized access.

Groups and Roles in Identity Management

Difference Between Groups and Roles:

•**Groups:** Collection of users with shared access rights.

•**Roles:** Assigned permissions based on job function.

💡 **Example:**

•**UNIX Groups:**

- •A user can be part of multiple groups (e.g., students, developers).
- •The newgrp command allows switching groups dynamically.

•**Role-Based Access Control (RBAC):**

- •System administrators assume different roles (e.g., **sysadmin, netadmin**).

🛠 **Security Considerations:**

•**Least Privilege Principle:** Users should have **only the access they need**.

•**Role Exploits:** Compromised accounts could **gain unintended privileges**.

Certificates and Identity Verification

- **Why Certificates Are Important?**
- They bind **cryptographic keys to an entity's identity**.
- Used for **secure communications and authentication**.

- 🔹 **Certificate Types:**
- **X.509 Certificates:** Used in HTTPS, SSL/TLS.
- **PGP Certificates:** Used in email encryption.
- 💡 **Example:**
- **Verisign Digital Certificates (1996):**
  - **Class 1** – Email verification.
  - **Class 2** – Personal identity verification.
  - **Class 3** – Background-checked high-security identity.

- 🛠 **Security Risks:**
- **Fake Certificates:** Attackers create certificates for fraudulent use.
- **Man-in-the-Middle Attacks:** Intercepted communications.

Identity and Anonymity on the Web

- **How Identity is Managed Online?**
- **Static Identifiers:** Fixed IP addresses, domain names.
- **Dynamic Identifiers:** DHCP-assigned IPs.
- 💡 **Example:**
- **Bootless University's DHCP Server:**
  - Assigns temporary IP addresses to student laptops.
- 🛠 **Security Issues:**
- **Spoofing:** Attackers pretend to be trusted users.
- **DNS Manipulation:** Redirecting users to malicious sites.

Anonymity on the Web

- **Definition:** Anonymity on the web allows users to communicate and browse without revealing their real identity. It is achieved by masking IP addresses and user data through anonymization techniques.
- **Purpose:**
- Protect privacy
- Prevent tracking and surveillance
- Enable free speech and whistleblowing
- Avoid targeted attacks
- **Key Mechanisms:**
- Proxy servers
- VPNs (Virtual Private Networks)
- Remailers (Email Anonymizers)
- Onion Routing & Tor

Email Anonymizers & Remailers

- **Email Anonymization:** This technique hides the sender’s real email address, preventing traceability.
- **2.1 Pseudonymous Remailers**
- Replace the sender’s email address with an alias (e.g., anon123@remailer.com)
- Store the mapping between real and pseudonymous addresses
- **Weakness:** If the remailer is compromised, anonymity is lost.
- **2.2 Cypherpunk Remailers (Type 1)**
- Remove headers and metadata before forwarding
- Use encryption to obscure message origins
- Messages are sent through multiple remailers for added anonymity
- **Weakness:** Can be vulnerable to traffic analysis

Cypherpunk remailer (also known as a mix network) works to anonymize email communication.

•**Layered Encryption**:
•The sender (Bob) encrypts the message multiple times, forming an "onion" structure.
•Each layer of encryption corresponds to a remailer in the chain.
•**Message Routing**:
•Bob first encrypts the message such that only the final recipient (Alice) can read it.
•He then adds another encryption layer for remailer 2, instructing it to send the inner message to Alice.
•Finally, he adds another layer for remailer 1, instructing it to forward the message to remailer 2.
•**Remailers**:
•**Remailer 1** receives the message, decrypts its layer, and sees that it must forward the remaining encrypted message to remailer 2.
•**Remailer 2** decrypts its layer, revealing that the final recipient is Alice, and sends it to her.
•Since Alice has the private key to decrypt the final message, she reads Bob's message.

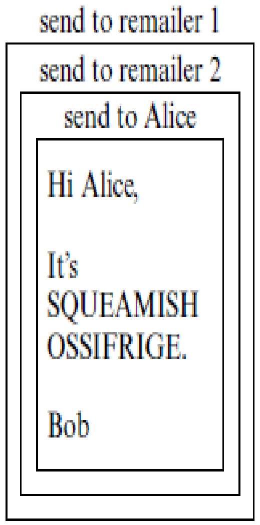


Figure 15–1 A message sent to a Cypherpunk remailer. Remailer 1 forwards the message to remailer 2, and remailer 2 sends it to Alice.

- **2.3 Mixmaster Remailers (Type 2)**
- Improve upon Cypherpunk remailers
- Standardize message sizes to prevent size-based tracking
- Introduce delays to prevent timing attacks
- **Strength:** Highly resistant to traffic analysis and replay attacks

Mixmaster remailer (a more advanced version of a Cypherpunk remailer) processes anonymous messages.

•**Layered Encryption (Onion Routing)**:

•The sender encrypts the message multiple times using a combination of **public-key encryption** (for remailers) and **symmetric encryption** (for efficiency).

•Each remailer in the chain can only decrypt its designated layer, ensuring no single entity knows both the sender and recipient.

•**Routing Process**:

•The **outermost layer** is encrypted with the **public key of remailer #1**, instructing it to forward the message to remailer #2.

•Inside this, another layer is encrypted with the **public key of remailer #2**, containing the recipient's address and actual message.

•Only the **last remailer** can see the final recipient's address

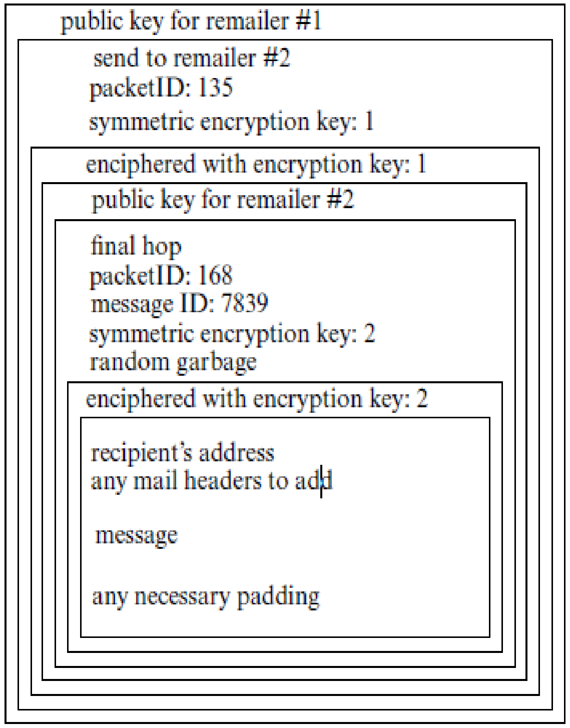
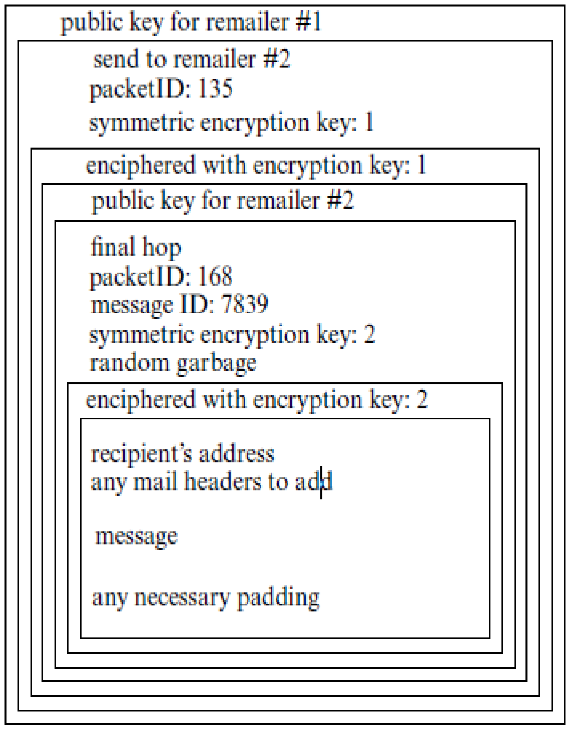

public key for remailer #1
send to remailer #2
packetID: 135
symmetric encryption key: 1
enciphered with encryption key: 1
public key for remailer #2
final hop
packetID: 168
message ID: 7839
symmetric encryption key: 2
random garbage
enciphered with encryption key: 2
recipient's address
any mail headers to add
message
any necessary padding

Figure 15–2 A Mixmaster message. This is a fragment of a multipart message sent through two remailers. Messages are enciphered using both a public key and symmetric key algorithm, and random garbage is added as well as padding. The recipient's address is visible only to the last remailer.

Mixmaster remailer (a more advanced version of a Cypherpunk remailer) processes anonymous messages.

- **Security Enhancements Over Cypherpunk Remailers**:
- **Packet IDs:** Each message fragment has a unique **packet ID** to prevent replay attacks (where an attacker resends an intercepted message).
- **Random Garbage & Padding:** Random data is added to prevent size-based analysis (traffic analysis).
- **Public Key & Symmetric Encryption:** Public key encryption is used to establish symmetric keys, making the decryption process more efficient.



public key for remailer #1
send to remailer #2
packetID: 135
symmetric encryption key: 1
enciphered with encryption key: 1
public key for remailer #2
final hop
packetID: 168
message ID: 7839
symmetric encryption key: 2
random garbage
enciphered with encryption key: 2
recipient's address
any mail headers to add
message
any necessary padding

Figure 15–2 A Mixmaster message. This is a fragment of a multipart message sent through two remailers. Messages are enciphered using both a public key and symmetric key algorithm, and random garbage is added as well as padding. The recipient's address is visible only to the last remailer.

Onion Routing & Tor

Concept: Onion routing protects user anonymity by encrypting data multiple times and routing it through a network of relays.

3.1 How Onion Routing Works

- The sender encrypts the message multiple times (like layers of an onion)
- Each relay decrypts only one layer and forwards the message
- The final relay (exit node) sends the original message to the destination
- **Key Advantage:** No single relay knows both the sender and destination

3.2 The Tor Network

- Tor (The Onion Router) is an implementation of onion routing
- Uses multiple relays to mask user identity
- Websites accessed through Tor end in .onion
- **Use Cases:**
    - Bypassing censorship
    - Whistleblowing and investigative journalism
    - Anonymous web browsing

- **3.3 Security Threats in Onion Routing**
- **Exit Node Monitoring:** The final relay can see unencrypted data
- **Traffic Analysis:** Adversaries monitoring network traffic can infer user activities
- **Malicious Nodes:** Some relays may be controlled by attackers

Security Threats & Attacks

- **1 Traffic Analysis Attacks**
- **Passive Attack:** Observing communication patterns without interference
- **Active Attack:** Injecting dummy messages to analyze responses
- **2 Replay Attacks**
- Attackers resend intercepted messages to analyze their path
- Mitigation: Remailers use unique message IDs to detect duplicates
- **3 Sybil Attacks**
- A single entity controls multiple nodes in a decentralized network
- Compromises anonymity by linking incoming and outgoing traffic
- **4 DNS and IP Tracking**
- Even when using Tor, DNS requests can leak identity
- VPNs and secure DNS resolvers help mitigate this risk

Ethical and Societal Implications of Anonymity

- **Benefits of Anonymity**
- **Privacy Protection:** Prevents excessive surveillance and tracking
- **Freedom of Speech:** Enables dissent in oppressive regimes
- **Whistleblowing:** Allows exposing corruption without retaliation
- **Security Against Targeted Attacks:** Protects activists, journalists, and researchers
- **Risks and Misuse**
- **Cybercrime:** Anonymity facilitates illicit activities like hacking, drug trade, and fraud
- **Misinformation and Harassment:** Anonymous trolls and false information spread rapidly
- **Legal & Ethical Concerns:** Striking a balance between privacy rights and law enforcement

Ethical and Societal Implications of Anonymity

3. Balancing Privacy and Security

- Governments and organizations need to balance privacy with security
- Implementation of ethical guidelines for anonymous communication
- Technologies like Tor and VPNs should be used responsibly

Case Study 1: Silk Road – The Dark Side of Anonymity

- **Overview:**
  Silk Road was an anonymous online marketplace on the **dark web** that facilitated illegal transactions, including drug sales, weapons trading, and hacking services. It relied on **Tor (The Onion Router)** for anonymity and **Bitcoin** for untraceable transactions.
- **How it Worked:**
- Users accessed Silk Road using the **Tor network**, which hid their IP addresses.
- Transactions were made using **Bitcoin**, making it difficult to trace buyers and sellers.
- The site's administrator, **Ross Ulbricht (alias Dread Pirate Roberts)**, used encryption techniques to stay anonymous.

Case Study 1: Silk Road – The Dark Side of Anonymity

- **How Authorities Caught the Owner:**
- **Traffic analysis**: Despite using Tor, authorities correlated his **forum posts** and **Bitcoin transactions** to his real identity.
- **Operational security flaws**: Ulbricht once logged into an online forum without Tor, exposing his **real IP**.
- The FBI seized the website and **arrested Ulbricht in 2013**.

Case Study 2: Edward Snowden and Whistleblowing

- **Overview:**

  Edward Snowden, a former **NSA contractor**, leaked classified information in 2013 about global mass surveillance programs run by the **U.S. National Security Agency (NSA)** and other governments.
- **How He Used Anonymity:**
- Communicated using **secure, anonymous email services**.
- Used **Tor and VPNs** to hide his location.
- Released files to journalists via **encrypted channels**.

- **Outcome:**
- **Major revelations** about mass surveillance and data collection.
- **PRISM program** exposed (NSA collected data from Google, Facebook, Apple, etc.).
- Snowden fled the U.S. and is **now in exile in Russia**.
- **Key Takeaways: ✓ Anonymity protects whistleblowers and journalists**
  ✓ Encryption and anonymity tools empower free speech
  ✗ Governments actively track and prosecute anonymous informants

Unit No: 4 Unit Name: Systems Design

Control of Access

Introduction to Access Control Matrices

- The access control matrix is a theoretical model for access control, structuring the relationships between subjects (users, processes) and objects (files, devices). The matrix specifies what operations each subject can perform on various objects.
- **1.1 Structure of an Access Control Matrix**
- An access control matrix consists of:
- **Rows (Subjects)**: Represent users, processes, or devices that request access to objects.
- **Columns (Objects)**: Represent files, memory segments, devices, or other resources.
- **Entries (Permissions)**: Define the rights (read, write, execute, own, etc.) a subject has over an object.

Challenges in Implementing Access Control Matrices

- A straightforward implementation faces several issues:
- **Scalability**: A large number of subjects and objects leads to a huge, sparse matrix with many blank entries.
- **Management Complexity**: Adding/removing subjects and objects requires careful updates to the matrix.
- **Default Permissions**: Most users have similar access levels, leading to redundant entries.

Optimized Implementations of Access Control Matrices

- To overcome these challenges, several optimizations exist:
- **Access Control Lists (ACLs)**
- **Capabilities-based Mechanisms**
- **Locks and Keys (Cryptography-based controls)**
- **ORCON (Originator Controlled) Access Control Mechanisms**

Access Control Lists (ACLs)

ACLs optimize access control by storing permissions per object rather than maintaining a global matrix.

3.1 Definition

An **Access Control List (ACL)** for an object **O** is a set of pairs:

ACL(O) = {(Subject1, Rights1), (Subject2, Rights2), ...}

Each subject has defined rights over the object. If a subject is not listed, they have no access unless wildcard entries provide default rights.

	File1	File2	Process1	Process2
Process1	R/W/O	R	R/W/X/O	W
Process2	A	R/O	R	R/W/X/O

ACL Implementation in Operating Systems

- Operating systems use different approaches to manage ACLs effectively.
- **4.1 UNIX ACL Implementation**
- UNIX-based systems use a simplified form of ACLs based on three categories:
- **Owner**: User who created the file.
- **Group**: Users belonging to the file's group.
- **Others**: Everyone else.

Example: If user bishop creates a file with read/write access for himself, read access for the group, and no access for others, the permissions are represented as:
rw- r-- ---

- In UNIX, permissions are represented as triplets:

Owner	Group	Others
rw-	r--	---

4.2 Extended ACLs in UNIX/Linux

- To overcome limitations of traditional UNIX permissions, extended ACLs allow fine-grained control.
- **Example:**
- user::rw-
- user:skyler:rwx
- user:sage:r
- group::rw-
- group:child:rx
- mask::rw-
- other::r-

•skyler can read/write.
•sage, a member of family, can read.
•steven, in group child, can only read.

5. Managing ACLs

5.1 Who Can Modify an ACL?

•The **owner** typically has modification rights.

•Some systems allow rights transfer (e.g., **System R**, where the grant option allows permission delegation).

5.2 Do ACLs Apply to Privileged Users?

•Some systems **bypass ACLs** for administrators (root in UNIX, Administrator in Windows).

•Others enforce ACLs **even on privileged users** (e.g., Solaris UNIX applies full ACLs to root).

5.3 Groups and Wildcards in ACLs

•**Groups** simplify ACLs by grouping users together.

•**Wildcards** provide flexibility (e.g., allowing access for all users except one).

Example: ACLs with Groups and Wildcards in AIX and UNICOS

- Attributes
- :base permissions
- ❑ owner(bishop): rw-
- ❑ group(sys): r–
- ❑ others:
- ---extended permissions enabled
- ❑ specify rw- u:holly
- ❑ permit -w- u:heidi, g=sys
- ❑ permit rw- u:matt
- ❑ deny -w- u:holly, g=faculty

D Y PATIL DEEMED TO BE UNIVERSITY RAMRAO ADIK INSTITUTE OF TECHNOLOGY NAVI MUMBAI

Unit No: 4 Unit Name: Systems Design

Information Flow Control

Introduction to Information Flow

- nformation flow controls regulate how data moves within a system to maintain security.
- Information flow policies are crucial for preventing unauthorized access, data leakage, or corruption.
- Security mechanisms define constraints to ensure safe and controlled data transfer between different components.
- Ensuring proper information flow helps maintain both:
  - **Confidentiality**: Ensuring sensitive data does not leak to unauthorized users.
  - **Integrity**: Ensuring data is not altered by unauthorized processes.

Basics and Background

- two primary objectives:
  - **Confidentiality**: Preventing unauthorized access to sensitive data.
  - **Integrity**: Ensuring data modifications occur only by trusted sources.
- Example:
  - **Bell-LaPadula Model:** Focuses on confidentiality (No Read Up, No Write Down).
  - **Biba Model:** Focuses on integrity (No Write Up, No Read Down).
- Security models inherently enforce a**Basics and Background**
  - n information flow policy to manage data movements effectively.

Entropy-Based Analysis

- **Definition:** Information flows when the uncertainty (entropy) of one variable is reduced by another.
- **Mathematical Representation:** Command sequence c causes information flow from x to y if:
    - **$H(xs \mid yt) < H(xs \mid ys)$** (where H represents entropy)
- **Examples:**
    - y := x; → Full information transfer from x to y.
    - y := x + z; → Partial transfer, dependent on z.
    - if x = 1 then y := 0 else y := 1; → Implicit flow, as y reveals information about x.

Information Flow Models

- **Lattice-Based Models:**
  - Defines a strict hierarchical flow of information.
  - Example: **Bell-LaPadula (Confidentiality)** and **Biba (Integrity)** models.
- **Nonlattice Models:**
  - Some policies do not form a strict hierarchy.
  - Example: Organizations where departments collaborate with overlapping but non-hierarchical access.
- **Nontransitive Models:**
  - Transitivity does not always apply (A trusting B and B trusting C does not mean A trusts C).
  - Example: Security in social networks or access controls with specific trust boundaries.

Static Mechanisms (Compile-Time Analysis)

- **Purpose:** Detect policy violations before execution.
- **Key Techniques:**
  - **Data Flow Analysis:** Tracks how data propagates in a program.
  - **Type-Checking:** Ensures assignments respect security constraints.
  - **Lattice-Based Enforcement:** Determines permissible data movements.
- **Advantages:**
  - Prevents security breaches at an early stage.
  - Enforces strict policy compliance.
- **Limitations:**
  - May reject safe programs due to conservative rules.
  - Cannot handle dynamic security decisions effectively.

Program Statements & Security

- **Assignment Statements:**
  - Ensures security constraints are met in value assignments.
  - Example: x := y + z requires lub{y, z} ≤ x.
- **Conditional Statements:**
  - Branching statements may leak information indirectly.
  - Example: if x = 1 then y := 0 else y := 1; (reveals information about x).
- **Iterative Statements:**
  - Loop execution can leak information via execution time.
  - Example: while x > 0 do y := y + 1; (loops reveal x's initial value).

Dynamic Mechanisms (Runtime Analysis)

- **Detects violations during execution rather than compile-time.**
- **Fenton's Data Mark Machine:**
  - Uses security labels for variables and program counters.
  - Ensures runtime compliance with security policies.
- **Challenges:**
  - Implicit flows are harder to track dynamically.
  - Performance overhead due to continuous checking.
- **Example of a Dynamic Flow Violation:**
  - A high-security variable influences the program execution path, affecting lower-security variables.

Variable Classes and Information Flow

- **Security classifications for variables:** Ensures controlled flow of data.
- **Static vs. Dynamic Classification:**
    - **Static:** Fixed security level throughout execution.
    - **Dynamic:** Security level changes based on operations performed.
- **Example of Dynamic Classification:**
    - Variable z starts as Low security.
    - After being assigned a value from High, z inherits a High security label.
    - Unauthorized assignments are blocked to prevent data leaks.

Integrity Mechanisms

- **Ensures data is modified only by trusted sources.**
- **Biba Model:**
  - Prevents high-integrity data from being tainted by low-integrity sources.
  - Example: Prevents untrusted user inputs from modifying sensitive system logs.
- **Application in Secure Systems:**
  - Database systems enforce integrity constraints.
  - Secure communication protocols ensure messages are not altered by unauthorized sources.

Real-World Information Flow Controls

- **Privacy in Android Apps:**
  - Many apps collect and share user data beyond their intended purpose.
  - **TaintDroid:** A tool that tracks information flow in Android applications.
  - Example: Prevents apps from sending GPS data to unauthorized servers.
- **Firewalls and Network Controls:**
  - Restrict unauthorized data exchange between networks.
  - Example: Corporate firewalls prevent confidential data from being exfiltrated to untrusted networks.
  - Deep packet inspection techniques detect and block potential data leak

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Thank You