# Introduction to Database Security

# Database Security

- **Definition**: The practice of protecting databases against unauthorized access, corruption, and theft.
- **Importance**: Keeps sensitive data secure and ensures it is available and accurate for authorized users.

# Advantages of Databases

- **Shared Access:** Multiple users can access a single, centralized set of data.
- **Controlled Access:** Only authorized users can modify data.
- **Consistency**: Changes affect all users, avoiding discrepancies.
- **Reduced Redundancy:** No need for individual copies of the same data

# Key Security Requirements

- **Physical Integrity:** Protection from physical issues like power failures.
- **Logical Integrity:** Ensures changes in data do not impact unrelated fields.
- **Element integrity:** The data contained in each element are accurate.
- **Auditability:** Tracks who accessed or modified data.
- **Access Control:** Defines who can view or change specific data.
- **User authentication:** Every user is positively identified, both for the audit trail and for permission to access certain data.
- **Availability:** Data is available to authorized users whenever needed.

# Integrity of the Database

**Database integrity** means ensuring the **data in a database is accurate and trustworthy.**

- For a database to be reliable, **only authorized people** should be allowed to make changes, and the data must be protected from any damage or corruption, whether caused by hacking, power failures, or physical disasters like fire.

Two main problems can affect a database's integrity:

**1.Complete database damage**, like when the storage device breaks.

**2.Individual data issues**, such as when certain pieces of data become unreadable.

# Cont..

- To protect against these problems, **regular backups are essential**. These backups act as a safety net, so if something goes wrong, the data can be restored.
- Additionally, databases keep a **log of all transactions** (a record of every action, like account updates or course registrations). If something fails, the system can use these logs to restore the database to its last good state without losing important recent updates, allowing things to continue without disruption.

# Element Integrity

- Element integrity refers to **the accuracy and correctness of individual pieces of data** in a database.
- It is important because the database should always store correct and reliable information.
- **Authorized users** are responsible for entering accurate data, **but mistakes can still happen**, whether by users or automated systems.

To help catch and correct these errors, databases use three main techniques:

1. **Field Checks**: These ensure that the data entered is valid. For example, a field may require a number or a specific set of characters, and the system checks that the entered data follows these rules.

2. **Access Control**: This ensures that only authorized individuals can update or change certain data fields, preventing unauthorized or accidental modifications.

3. **Change Logs**: Every change made to the database is recorded in a log, allowing administrators to review and undo any incorrect changes.

# Auditability

- Auditability means **keeping a record of all actions** (reading or writing data) performed on a database.
- This record, known as an **audit trail**, helps administrators track who accessed or changed data and when.
- This is **useful for maintaining data integrity** and identifying who made specific changes if something goes wrong.
- A key benefit of auditability is that **it can help reveal patterns over time**, like solving a mystery by gathering clues. Even if each individual action doesn't reveal much, looking at them together can give a clearer picture of what data was accessed.

# Cont..

- However, **auditing every tiny action**, like viewing or changing specific records or fields, can be challenging and overwhelming for many systems.
- Sometimes, even if a user performs an operation (like asking for a count of failed students), they don't directly see the data (like the grades themselves), but their **action still gets recorded in the audit trail**.
- This can make the audit logs tricky, as they might **show more data being accessed than what the user actually learned.**

# Access control

Access control in databases refers to managing who is allowed to access different parts of the data.

1. **User Privileges**: Databases store data for different groups of users, and **not everyone should see everything.** For example, general data may be available to all employees, but only HR should access salary details, and only marketing can view sales data.

2. **Centralized Data**: Databases are useful because they store all data in one place, making it easier to manage. However, **controlling who can access which parts of the data is crucial.**

**3. Database Administrator's Role**: The **database administrator (DBA) is in charge of setting rules about who can access what data**. These rules can be very specific, down to the level of individual data fields or records. The database system enforces these rules, allowing access to authorized data and blocking unauthorized access.

**4. Different Access Types**: People can have **different levels of access.** Some may only be able to read data, while others can change or delete it. Some might be allowed to modify the entire database structure, while others can only add or delete certain records.

**5. Complexity of Access Control**: **Controlling access in databases is more complex than in operating systems. In an operating system**, files are separate from each other. But in a database, different data fields are connected, and reading one piece of data could potentially allow someone to figure out other pieces of data. **This is called "inference.“**

**6. Inference Problem**: Sometimes, **even without direct access to sensitive data, users can figure out protected information by analyzing other data they do have access to**. Preventing this can be tricky, as blocking too much access may hinder regular users from doing their work, and checking every query for inference risks can slow down the database's performance.

**7. Granularity**: Databases have a lot more detailed access control compared to operating systems. In an operating system, you control access to whole files. In **databases, access control can apply to much smaller units of data, like individual fields in a record**. This finer control is harder to manage and can affect how efficiently the system works.

# User Authentication

- **What is it?**
  - A process to verify that users are who they claim to be.
- **How does DBMS authenticate users?**
  - It can require **passwords** and check things like the **time of access**.
- **Why extra authentication?**
  - Even though the **operating system (OS)** also authenticates users, the **DBMS doesn't fully trust** the OS.
- **DBMS handles its own checks:**
  - It ensures user identities are valid by **double-checking credentials itself.**

# Availability

- **What is it?**
  - Ensuring that the database is **always accessible** when users need it.
- **Why is it challenging?**
  - Sometimes multiple users **request the same data**, causing **delays** or **denial of access** to one user.
- **When unavailability happens:**
  - If the system is **being repaired, upgraded**, or overloaded with requests, users will notice.
- **Balance needed:**
  - The DBMS must ensure **all users get fair access** while protecting sensitive data, which isn't easy.

# Integrity / Confidentiality / Availability (Security Aspects)

**Integrity:**

•Ensures **data is accurate and consistent**.

•Data structure (tables) must follow rules—like records being linked correctly across tables.

**Confidentiality:**

•Protects **sensitive data** from being accessed by unauthorized users.

•Access can be **direct** (you request specific records) or **indirect** (you infer information from multiple records).

•**Indirect access is harder to control** and needs special care.

**Availability:**

•Databases must be available to users as needed, but **availability can conflict with confidentiality**.

•Protecting data while keeping it accessible to the right users is a constant **balancing act**.

# Reliability and Integrity

**1. Reliability:**

- A **reliable system** runs for **long periods without failure.**
- Users expect the DBMS to **protect data from loss or corruption**.
- Even with controls, authorized users might **unintentionally input incorrect** data.

**2. Three Dimensions of Integrity:**

- **Database Integrity:** Protects the entire database from damage, like disk failures or index corruption.
- **Element Integrity:** Ensures only **authorized users** can modify specific data elements.
- **Element Accuracy:** Prevents incorrect values from being entered through **checks and constraints**.

# Cont..

## Protection from the Operating System (OS):

The OS provides **basic protections**, such as:

- **Backups** to prevent data loss.
- **Access control** to restrict unauthorized access.
- **Integrity checks** during data read and write operations.
- **DBMS adds additional protections** beyond the OS.

**Cont..**

## Two-Phase Update Process

- **Solves** the problem of **data corruption** if a system fails during an update.

**Phases:**

### 1.Intent Phase (Preparation):

1. DBMS gathers data and locks resources, but **no changes are made** to the database.
2. If the system fails, the entire phase can be **restarted safely**.

### 2.Commit Phase (Permanent Changes):

1. A **commit flag** is set, indicating that changes are now final.
2. The actual data updates happen in this phase.
3. If a failure occurs during the commit, the system can **repeat the commit phase** safely.

**Example:**

- A department requests 50 boxes of paper clips.
- The DBMS follows all necessary steps (deducting inventory, charging the department, triggering reorder).
- If a failure occurs **before the commit**, the transaction can restart without errors.
- If it fails **during the commit**, it will **retry until all steps are completed.**

## Cont..

## Redundancy and Error Correction:

- **Redundancy:** Duplicate data (like **shadow fields**) are maintained for quick recovery if data gets corrupted.

- **Error Detection Codes:**
  - **Parity bits, Hamming codes,** or **cyclic redundancy checks (CRC)** ensure that the data remains accurate during storage and transmission.

**Cont..**

## Concurrency and Consistency (Multiuser Access):

### 1.concurrency:

1. Multiple users can access the same database at the same time without conflicts.
2. **Reads are safe**, but simultaneous updates can cause issues.

### 2.Example: Airline Reservation Conflict:

1. Two agents **try to assign the same seat to different passengers**.
2. Solution: **DBMS locks the seat after one agent selects it**, ensuring the other agent can't assign it.

### 3.Read-Write Conflicts:

1. If **one user updates data while another tries to read it**, the reader might get **incomplete data**.
2. **Solution:** DBMS **locks the data** during updates to prevent this issue.

# Database Disclosure Overview

## 1. More data is being collected and shared:

1. With modern technology, **organizations collect data** from many places (e.g., websites, stores).
2. **Data-sharing** across organizations is now common, unlike in the past when data stayed within individual entities.

## 2. Data can lead to inferences:

1. **People can infer patterns or connections from data**, which might be true or false (e.g., "Your friend likes frogs, so you must like frogs too").
2. These inferences can lead to **privacy issues** and **unintended data disclosure**.

# Cont..

## Sensitive Data

### 1.What is Sensitive Data?

1. Data that **should not be made public** or shared with unauthorized users.
2. Sensitivity depends on **the type of data** and **its context.**

### 2.Examples of Sensitivity Levels:

1. **Public Data:** E.g., a library catalog with no sensitive information.
2. **Completely Sensitive Data:** E.g., a defense database with **highly confidential information**.
3. **Mixed Sensitivity:** E.g., a **university database** with varying degrees of sensitive information (like name, drug use, financial aid).

# Factors That Make Data Sensitive

## 1.Inherently Sensitive:

1. The value itself is revealing (e.g., locations of missiles or a small-town barber’s income).

## 2.Sensitive Source:

1. The source must remain confidential (e.g., data from an informer whose identity is protected).

## 3.Declared Sensitive:

1. Data is labeled as sensitive by **owners or administrators** (e.g., classified military information or an anonymous donor's name).

## 4.Sensitive Attributes or Records:

1. Some fields or records are **entirely sensitive** (e.g., salary data in a personnel database).

## 5.Sensitive When Combined with Other Data:

1. Individual data points may be harmless alone but **sensitive when combined** (e.g., longitude and latitude revealing a secret location).

## Cont..

## Access Control for Sensitive Data

### 1. Direct vs. Indirect Access:

1. **Direct access:** Requesting specific data (e.g., retrieving a student's name).
2. **Indirect access:** Inferring information from multiple records (e.g., guessing drug use based on other details).

### 2. Access Restrictions by Sensitivity Levels:

1. Some users may have **partial access** (e.g., only to student names).
2. Others may need access to **multiple sensitive fields**, but **no one should have access to everything**.

### 3. Goal of Access Control:

1. Ensure that **users only access** the data they are **authorized to see**.
2. **Prevent unauthorized disclosure** of sensitive data.

# Types of Disclosures:

1. **Exact Data Disclosure**:
   - Sensitive data values can accidentally or intentionally be revealed.
   - Even unintended access to sensitive data compromises security.
2. **Bounds Disclosure**:
   - Revealing a range (e.g., salary between $50,000 and $82,000) can disclose sensitive information.
   - Narrowing down ranges iteratively allows guessing precise values.
3. **Negative Result Disclosure**:
   - Learning what is *not* true can also be revealing (e.g., knowing someone has at least one felony conviction implies they have a criminal record).
4. **Existence Disclosure**:
   - Just discovering that some data exists (like call monitoring logs) can be sensitive.
5. **Probable Value Disclosure**:
   - Probabilities can hint at sensitive values (e.g., guessing party registration based on query patterns).

# Inference Attacks:

**1. Direct Inference**:

Sensitive data can be inferred from non-sensitive data through clever queries.

**2. Direct Attack**:

Queries crafted to target specific values or individuals (e.g., asking for people with certain attributes).

**3. Arithmetic Inference**:

Attackers use released statistical summaries (like sums and averages) to deduce individual data.

**4. Tracker Attack**:

Attackers manipulate queries and retrieve partial data to derive hidden information.

**5.Linear System Vulnerability**:

A system of equations can reveal hidden values by solving logical or numerical relationships.

# Example

| Name | Sex | Race | Aid | Fines | Drugs | Dorm |
|---|---|---|---|---|---|---|
| Adams | M | C | 5000 | 45. | 1 | Holmes |
| Bailey | M | B | 0 | 0. | 0 | Grey |
| Chin | F | A | 3000 | 20. | 0 | West |
| Dewitt | M | B | 1000 | 35. | 3 | Grey |
| Earhart | F | C | 2000 | 95. | 1 | Holmes |
| Fein | F | C | 1000 | 15. | 0 | West |
| Groff | M | C | 4000 | 0. | 3 | West |
| Hill | F | B | 5000 | 10. | 2 | Holmes |
| Koch | F | C | 0 | 0. | 1 | West |
| Liu | F | A | 0 | 10. | 2 | Grey |
| Majors | M | C | 2000 | 0. | 2 | Grey |

# Direct attack:

```
List NAME where
        SEX=M ∧ DRUGS=1
```

# An attack by sum tries to infer a value from a reported sum.

| | Holmes | Grey | West | Total |
|---|---|---|---|---|
| M | 5000 | 3000 | 4000 | 12000 |
| F | 7000 | 0 | 4000 | 11000 |
| Total | 12000 | 3000 | 8000 | 23000 |

**TABLE 7-8** Table Showing Negative Result

This seemingly innocent report reveals that no female living in Grey is receiving financial aid. Thus, we can infer that any female living in Grey (such as Liu) is certainly not receiving financial aid.

# Linear System Vulnerability

$$
\begin{aligned}
q_1 &= c_1 + c_2 + c_3 + c_4 + c_5 \\
q_2 &= c_1 + c_2 \quad\quad + c_4 \\
q_3 &= \quad\quad\quad c_3 + c_4 \\
q_4 &= \quad\quad\quad\quad\quad c_4 + c_5 \\
q_5 &= \quad\quad c_2 \quad\quad\quad\quad + c_5
\end{aligned}
$$

To see how, use basic algebra to note that $q_1 - q_2 = c_3 + c_5$, and $q_3 - q_4 = c_3 - c_5$. Then,
subtracting these two equations, we obtain $c_5 = ((q_1 - q_2) - (q_3 - q_4))/2$.
Once we know $c_5$,
we can derive the others.

# Aggregation & Metadata

**Aggregation Problem**:

- Combining multiple non-sensitive data points from different sources can reveal sensitive information.
- This is difficult to prevent as aggregation can occur outside the database system.

**Metadata Issues**:

- Digital files contain hidden attributes (e.g., photo tags, location data) that may unintentionally disclose private information.
- Geotagging, for example, can reveal someone's location even when they didn't intend it.

**Tracking Devices**:

- Devices like phones, GPS, and RFID tags can create a detailed log of someone's movements.

**Anonymized Data Risks**:

- Even when data is anonymized (e.g., replacing names with IDs), patterns can reveal identities (as shown in the AOL search data case).

- **Takeaway:**
  - Sensitive data can be disclosed not only directly but through subtle means like inference, metadata, or aggregation.
  - Effective security requires not just limiting access to sensitive data but also controlling indirect ways of inferring it.

# Preventing Disclosure: Data Suppression and Modification

**Challenges of Controlling Inference and Aggregation**:

- No perfect solutions exist to prevent unauthorized inferences from data.
- Strategies aim to either limit the **queries** users make or restrict the **data** provided in response.

# Methods to Prevent Disclosure:

- **Suppress Sensitive Information**:
  - Block access to obviously sensitive data.
  - Often leads to withholding too much data, reducing the usefulness of the database.
- **Track User Knowledge**:
  - Monitor what a user has accessed to limit further sensitive information.
  - Expensive and complex, especially with users using multiple IDs or colluding.
- **Disguise Data with Perturbation or Rounding**:
  - Modify values slightly by adding small errors (perturbation) or rounding.
  - Helps prevent exact statistical inferences but may yield imprecise results.

# Security vs. Precision:

- **Security**: Maximize privacy by withholding sensitive data.
- **Precision**: Share as much non-sensitive data as possible.

- Balancing these goals is difficult.
- Example: A query for grades of students using drugs may not compromise privacy, but rejecting all such queries limits valid research.

# Techniques for Data Suppression and Modification:

- **Limited Response Suppression**:
  - Suppress low-frequency data (e.g., counts of 1) to protect privacy.
  - Requires additional suppression to prevent values from being inferred by subtraction from totals.
- **Combining Values or Presenting Ranges**:
  - Combine sensitive values (e.g., drug use levels) to make them less identifiable.
  - Release data in ranges (e.g., financial aid of $0–1999) instead of exact values.
- **Random Sampling**:
  - Results are based on a **sample** of the database, not the whole.
  - Prevents attackers from making inferences based on repeated queries.
- **Blocking Small Sample Sizes**:
  - Avoid releasing data if too few individuals dominate a category (e.g., if 1 person represents 100% of the data).
  - Helps prevent private information from being revealed.
- **Random Data Perturbation**:
  - Add small random errors to data values to prevent exact inferences.
  - Works well for aggregations, as statistical trends remain accurate.
- **Swapping Data**:
  - Exchange certain fields (e.g., swapping gender between two records) to protect privacy.
  - Data stays useful, but exact values are not guaranteed to be accurate.

## Query Analysis for Security:

- **Analyzing Queries**: Track the context and history of a user's queries to identify potential inference risks.
- Example: A query that seems harmless alone may reveal sensitive information when combined with previous queries.

## Finding the Balance:

- **Binary Access Control** (yes/no) isn't practical for databases—either it restricts access too much or risks too much exposure.
- **Middle Ground**: Database systems try to balance security and usability through nuanced controls, avoiding extremes.

# Part 2- Cloud Computing Security

- Introduction to Cloud Computing,
- Service and Deployment Models,
- Risk Analysis,
- Cloud as a Security Control,
- Cloud Security Tools and Techniques,
- Cloud Identity Management,
- Securing IaaS.

# 1. Introduction to cloud computing

- it is a new way of providing services by using technology.
- cloud computing is a model “for enabling convenient, on-demand network access to a shared pool of configurable computing resources.”
- Cloud consists of networks, servers, storage, applications, and services that are connected in a loose and easily reconfigurable way.
- **Cloud computing implies the export of processors, storage, applications, or other resources. Sharing resources increases security risk.**

## The cloud has five defining characteristics:

- On-demand self-service.

  If you are a cloud customer, you can automatically ask for computing resources (such as server time and network storage) **as you need them**.

- Broad network access.

  You can access these services with a variety of technologies, such as mobile phones, laptops, desktops, and mainframe computers.

- Resource pooling.

  The cloud provider can put together a large number of multiple and varied resources to provide your requested services.

  This "multitenant model" permits a single resource (or collection of resources) to be accessed by multiple customers, and a particular resource (such as storage, processing or memory) can be assigned and reassigned dynamically, according to the customers' demands.

- *Rapid elasticity*.

  Services can quickly and automatically be scaled up or down to meet a customer's need.

  To the customer, the system's capabilities appear to be unlimited.

- *Measured service*. Like water, gas, or telephone service, use of cloud services and resources can be monitored, controlled, and reported to both provider and customer.

# 2. Service Models

- A cloud can be configured in many ways, but there are three basic models with which clouds provide services

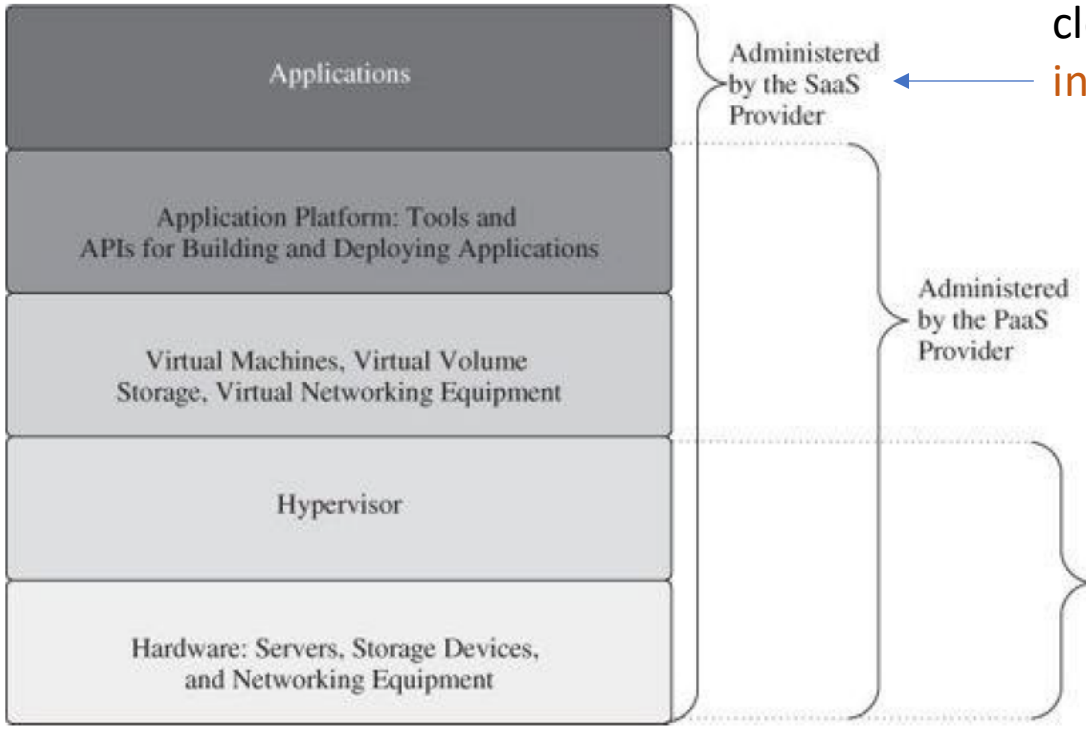



**FIGURE 8-1** Cloud Service Models

cloud provider gives a customer access to applications running in the cloud. **Software as a service: applications in the cloud -**

the customer has his or her own applications, but the cloud affords the languages and tools for creating them. **Platform as a service: languages and tools to support application development in the cloud**

**Infrastructure as a Service:** the cloud offers processing, storage, networks, and other computing resources that enable customers to run any kind of software.

# Deployment Models

- There are many different definitions of clouds, and many ways of describing how clouds are deployed.
- Often, four basic offerings are described by cloud providers: **private clouds, community clouds, public clouds, and hybrid clouds.**

- A **private cloud** has infrastructure that is operated exclusively by and for the organization that owns it, but cloud management may be contracted out to a third party.
- A **community cloud** is shared by several organizations and is usually intended to accomplish a shared goal.
  - For instance, collaborators in a community cloud must agree on its security requirements, policies, and mission.
  - It, too, may farm out cloud management to another organization.
- A **public cloud**, available to the general public, is owned by an organization that sells cloud services.
- A **hybrid cloud** is composed of two or more types of clouds, connected by technology that enables **data and applications to be moved around the infrastructure to balance loads among clouds**.

# 3. Risk Analysis

- Before moving functionality or data to a cloud, it is important to consider pros and cons.
- Moving to a cloud can be difficult and expensive, and it can be equally expensive to undo.
- While every cloud offering presents its own set of risks and benefits, a number of guidelines can help you understand whether your functions and data should be migrated to a cloud environment, as well as which cloud offerings will be most likely to meet your security needs.

# Risk analysis should be a part of any major security decision, including a move to cloud services.

**1. Identify assets.**

Moving to a cloud service generally means moving functionality and data.

It is important that you document every function and data type that might move to the cloud service, since it's easy to lose track and miss something important.

**2. Determine vulnerabilities.**

When considering cloud services, be sure to consider cloud-specific vulnerabilities.

These will generally stem from having to access the system through an Internet connection, sharing hardware and networks with potential adversaries, and trusting a cloud provider.

## 3. Estimate likelihood of exploitation.

Many vulnerabilities will be either more or less difficult to exploit in a cloud environment, as well as across different cloud service models and providers.

## 4. Compute expected loss.

Evaluate how easily vulnerabilities can be exploited in a cloud environment compared to on-premises solutions.

Consider whether a typical cloud provider can respond to attacks (e.g., DDoS) more effectively than your organization.

## 5. Survey and select new controls.

Identify necessary security controls for managing risks in the cloud, including:

- Data encryption needs.
- Logging capabilities from the provider.
- Authentication and access control options.

## 6. Project Savings

Cost-Benefit Analysis: While moving to cloud services is often justified by projected cost savings, hidden costs may arise.

Understanding Total Costs: Evaluate all potential expenses, including new security controls, to avoid unexpected financial burdens.

# 4. Cloud as a Security Control

- While moving data and functionality to the cloud does have its risks, cloud services can be **valuable security tools** in a number of ways.
- The most obvious is that cloud services are often excellent at **mitigating single points of failure.**

## Mitigating Single Points of Failure

- Geographic Diversity: Having multiple data centers reduces risks from localized threats (natural disasters, fires, internet outages).
- Choose secondary data centers far from primary ones to minimize shared risks.

## Cloud RAID Concept

- Innovative Approach: Researchers at Cornell University proposed a method to combat vendor lock-in by engaging multiple cloud storage providers.
- **Redundant Array of Cloud Storage (RACS): Treats multiple cloud providers as a single RAID array.**
- Maintains redundancy to **recreate data if one provider fails**.
- Estimated cost **increase of only 11%** compared to traditional storage solutions.

## Platform Diversity

**Varied Vulnerabilities**: **Different operating systems, applications, and protocols** used by cloud providers can **reduce the likelihood of simultaneous attacks** on both your systems and theirs.

**Decreased Risk:** The diversity in platforms helps in minimizing common vulnerabilities.

## Infrastructure Diversity

**Diverse Vulnerabilities**: Differences in **hardware, network configuration, security controls**, and quality of security staff between your organization and the cloud provider.

**Enhanced Security Posture**: **This diversity can provide additional layers of security** against attacks.

## Additional Security Operations in the Cloud

**Email Filtering**: Cloud **providers can filter spam and dangerous attachments** before reaching user inboxes.

**DDoS Protection**: Cloud-based services **can absorb attack traffic and filter** malicious packets before they reach customer systems.

**Network Monitoring**: Cloud solutions can **handle log analysis efficiently**, alleviating resource constraints.

# Cloud Security Tools and Techniques

## 1. Introduction to Cloud Security

- **Definition**: Cloud security is the protection of cloud-based systems, data, and infrastructure from cyber threats.
- **Relation to Information Security**: While it shares principles with general information security, cloud security introduces unique challenges **due to shared resources.**

## 2. Unique Threat Vectors

- **Shared Resources**: **Risks arise from shared processing**, storage, and communication resources with potential adversaries.
- **Adapting Security Tools**: **Standard tools like encryption and network security** must be tailored for cloud environments.

## 3. Data Protection in the Cloud

- **Public Cloud Services**: Involves sending private data over the Internet and storing it on the provider's servers (SaaS, PaaS, IaaS).
- **Responsibility**: Users must **select cloud offerings that ensure data protection** from unauthorized access and modifications.

## 4. Securing Data in Transit

- **Encryption Protocols:** Use TLS for SaaS/PaaS services.
- For IaaS, consider additional methods like SSH and VPNs.
- **mutual authentication**, allowing the client and server to authenticate each other.

## 5. Cloud Storage Considerations

- **Data Sensitivity Assessment**: Determine encryption needs based on the sensitivity of stored data.
- **Access Control Requirements:** Understand sharing capabilities and their implications for sensitive information.

## 6.Compliance and Regulations

- **Export Controls**: Be aware of regulations affecting data flow across borders.
- **Provider Audits**: Ensure cloud providers comply with necessary regulations and standards.

## 7. Data Confidentiality Measures

- **Encryption Standards**: Utilize AES-256 with individual keys for users.
- **Key Management Strategy**: Implement a master key system to facilitate secure user key changes without re-encrypting large datasets.

## 8. Trust No One Philosophy (TNO)

- **Definition**: Providers should not have access to user encryption keys.
- Example - Lastpass: protect user data without storing decryption keys on their servers.

## 9. Data Loss Prevention (DLP)

- **Challenges with DLP in Cloud Environments**: Traditional DLP measures may be bypassed when accessing cloud services outside company networks.
- **Solutions**: Enforce VPN access for remote logins.
- **Deploy DLP solutions within** IaaS environments.

## 10. Cloud Application Security

- **Secure Software Development**: Follow best practices for application security in shared environments.
- **Common Threats**: Attacks on shared resources (e.g., SQL injection).
- Insecure APIs lead to vulnerabilities.

## 11. Logging and Incident Response

- **Importance of Logs**: Essential for detecting and investigating security incidents in cloud environments.
- **Challenges in Public Cloud**: Limited access to logs from SaaS/PaaS; IaaS offers more control over logging.

## 12. Best Practices for Incident Response

- **SLAs with Providers**: Include requirements for logging, evidence preservation, and incident notification.
- Ensure logs are sent to a separately for analysis and protection against intrusions.

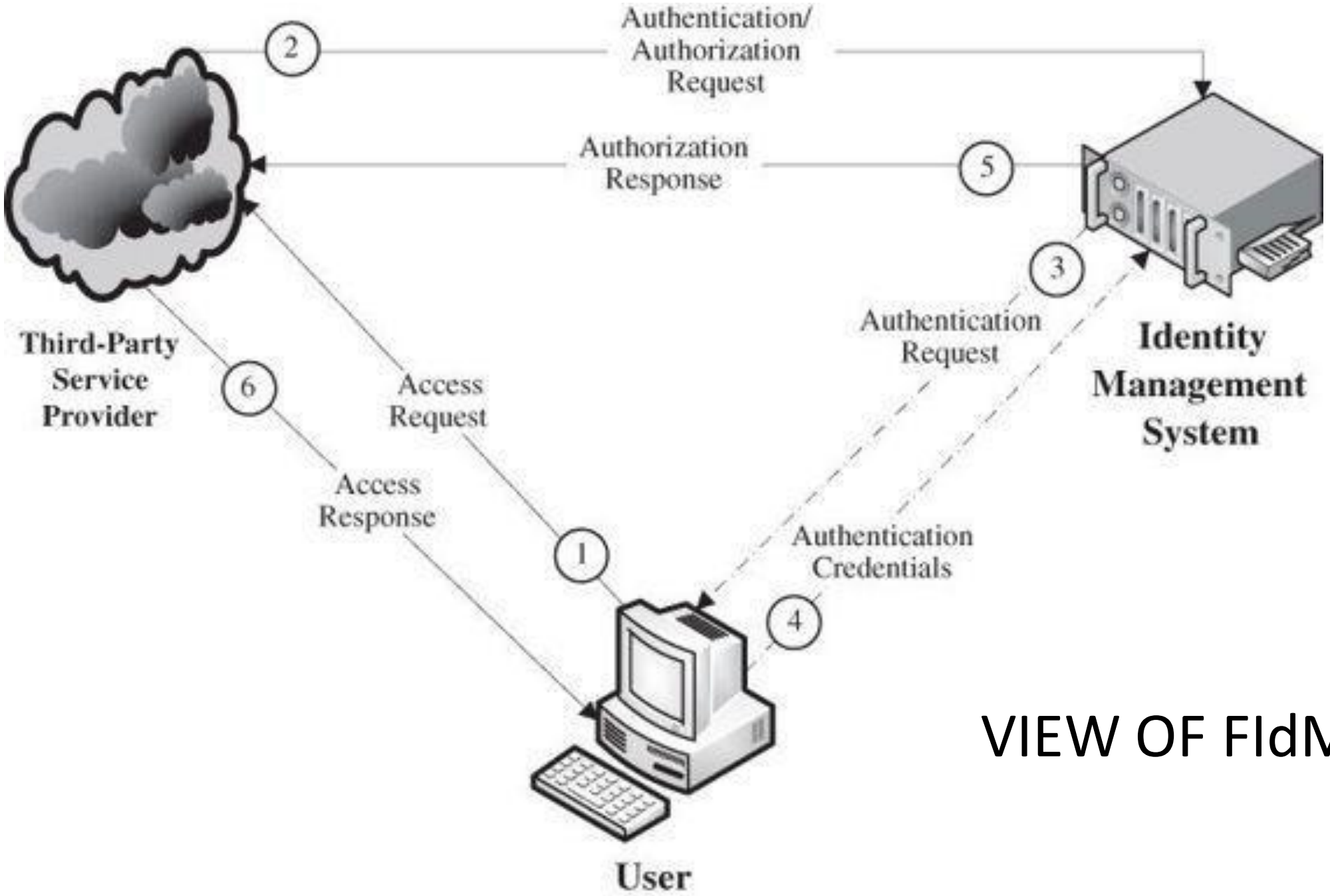
# Introduction to Cloud Identity Management

- **Definition**: Cloud Identity Management refers to the processes and technologies used to manage user identities and their access to cloud resources.
- **Importance:** As organizations migrate sensitive data to the cloud, **effective identity management is crucial** for authentication and authorization.

- **Challenges in Cloud Identity Management**
  - **Multiple Accounts**: Users often need separate accounts for different cloud services, increasing vulnerability.
  - **Password Management Issues**: Weak passwords and reuse across services heighten security risks.
  - Loss of Control: Organizations may struggle to manage user accounts effectively across various cloud providers.

- Risks of Individual Account Creation
  - **Security Vulnerabilities**: Increased chances of data breaches **due to poor password practices.**
  - **Administrative Burden:** Difficulty in managing user access when **employees leave or change roles.**

- **Risks of Shared Accounts**
  - **Increased Theft Risk**: Shared passwords can easily be compromised.
  - **Accountability Issues**: Hard to determine who accessed what data.
  - **Frequent Password Changes**: Necessity to change passwords whenever a user leaves or changes roles.
- **Federated Identity Management (FIdM)**
  - Definition: FIdM **allows identity information to be shared among multiple entities**, providing single sign-on (SSO) convenience.
  - How It Works: One **system maintains user identity information**.
  - Other systems query this central system for authentication.
- **Benefits of Federated Identity Management**
  - **Single Sign-On (SSO):** Users can access multiple services with one set of credentials.
  - **Enhanced Security Controls**: Organizations can enforce strong password policies and multifactor authentication.
  - ***Simplified User Lifecycle Management:*** Centralized control over user accounts ensures efficient provisioning and de-provisioning.

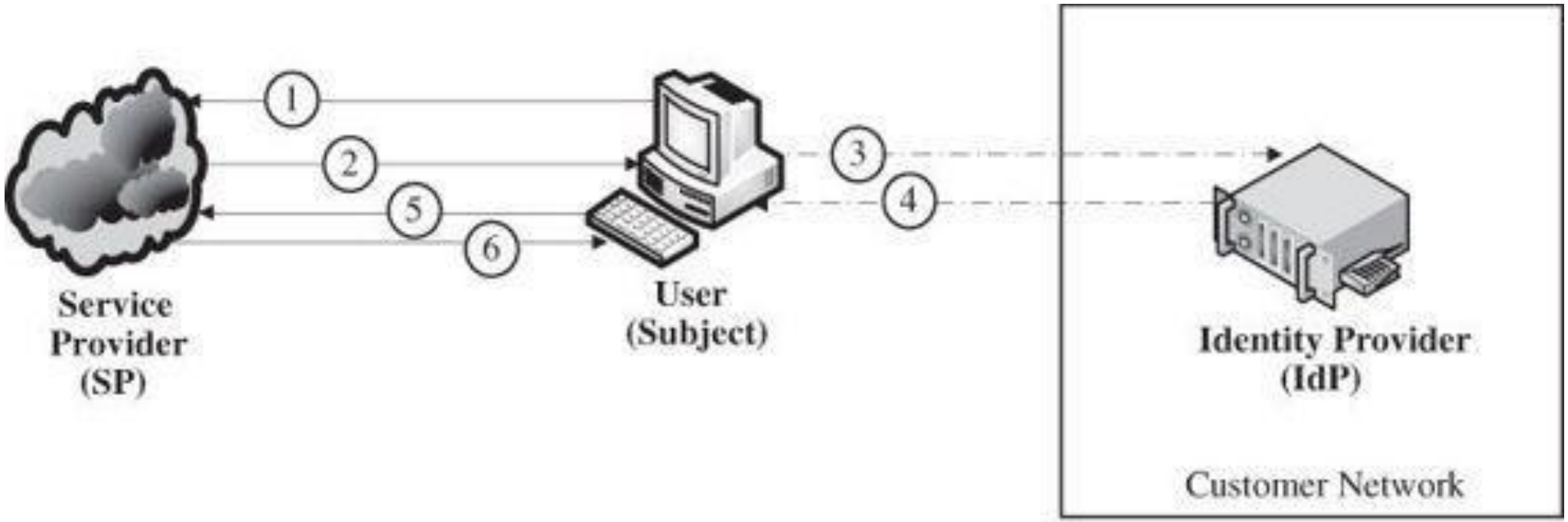

Authentication/
Authorization
Request
2
Authorization
Response
5
3
Authentication
Request
Identity
Management
System
Third-Party
Service
Provider
6
Access
Request
Access
Response
1
Authentication
Credentials
4
User

VIEW OF FIdM

# Security Assertion Markup Language (SAML)

- Definition: **SAML is an XML-based standard for securely exchanging user identity information between systems.**
- Key Components: **Service Provider (SP) / Relying Party**:: The application requesting identity information.
- The **Subject**: The entity, be it user or system, that is attempting to log in to the SP
- Identity Provider (IdP)/ **Asserting Party**: The system authenticating the user and providing identity assertions.

- SAML Authentication Flow
  1. **User attempts to access SP.**
  2. **SP redirects user to IdP for authentication.**
  3. **IdP authenticates the user and sends back a signed message with identity attributes.**
  4. **SP grants access based on assertions received.**
- Types of SAML Assertions
  - **Authentication Assertion:** Confirms the user was authenticated at a specific time.
  - **Attribute Assertion**: Provides additional attributes about the user (e.g., roles, permissions).
  - **Authorization Decision Assertion**: Indicates whether access to a resource is granted or denied.




**SAML Authentication Process**

1. Subject navigates to the SP site for login
2. SP sends the subject's browser an authentication request
3. Browser relays the authentication request to the IdP
4. IdP attempts to authenticate the subject, then returns the authentication response to the browser
5. Browser relays the authentication response to the SP
6. SP reads the authentication response and, if the user is authorized, logs the user in with the privileges the IdP specified

# OAuth (Open Authorization) for Authentication

## OAuth does not exchange identity information, just authorization

- **Integration with SAML**: OAuth can be combined with SAML for native applications, allowing seamless identity management across platforms.
- **OpenID Connect (OIDC)**: A newer standard built on OAuth that supports broader use cases beyond enterprise applications.

- **Advantages of OpenID Connect**
  - **User-Friendly Authentication**: Allows users to log into multiple services using existing credentials (e.g., Google).
  - **Support for Native Applications**: OIDC is designed to work well with both web and mobile applications.
- **Best Practices for Cloud Identity Management**
  - **Implement Strong Authentication Mechanisms**: Utilize multifactor authentication and enforce password complexity requirements.
  - **Regularly Review Access Permissions:** Conduct audits to ensure that **users have appropriate access levels.**
  - **Adopt Federated Identity Solutions**: Use FIdM to streamline access management across multiple cloud providers.

# Securing IaaS

- **Introduction to IaaS**
  - Infrastructure as a Service (IaaS) allows scalable server infrastructure.
  - Ideal for dynamic needs of MMO games.
  - Pay only for what you use; rapid elasticity to handle varying player demand.
- **Key Features of IaaS**
  - **Virtualization**: Leverages hypervisors to manage multiple VMs.
  - **Cloud Management**: Monitors, provisions, and manages workloads.
  - **Flexibility**: Quickly adjust resources based on player activity.

- **Public IaaS vs. Private Network Security**
  - **Shared Infrastructure**: New threats arise from shared resources.
  - **Access Methods**: More access points (APIs, consoles) than traditional setups.
  - **Deployment Ease**: Easy to create new VMs and networks, requiring stricter security.
- **Addressing Shared Infrastructure Threats**
  - **Shared Storage Risks**: Data remains on disks until overwritten.
    - **Mitigation**: **Use encryption for sensitive data.**
  - **Shared Network Risks**: Network traffic could be intercepted.
    - **Mitigation**: **Encrypt traffic with TLS, SSH, or VPN.**
- **Securing Host Access**
  - **Protect** management interfaces (**web consoles, APIs**).
- **Authentication Strategies**:
  - Use **multifactor a**uthentication.
  - Avoid shared accounts; **enforce least privilege.**
  - Use **OAuth for API access**.

## Virtual Infrastructure Best Practices

- Create specialized VMs for different functions (e.g., FTP server).
- **Hardening VMs**:
  - Disable unnecessary services and privileges.
  - Implement application whitelisting.
  - Configure host-based firewalls.
- **Network Segregation**
  - Utilize private network enclaves for different system functions.
  - Protect enclaves with strict firewall rules.
  - Use application proxy servers to manage external access.
- **Monitoring and Logging**
  - Limit SSH and screen-sharing access to trusted IPs.
  - Collect logs from VMs; do not store them on the same infrastructure.
  - Analyze logs for failed login attempts to enhance security.