Contents

1 Overview

1.1 Computer Security Concepts

1.1.0.1 Essential network and Security Requirements

C.I.A.

Definition by NIST : *Measures and controls that ensure confidentiality, integrity, and availability of information system assetsincluding hardware, software, firmware, and information being processed, stored, and communicated*

Confidentiality It can be divided into two kind :

Data Confidentiality Assures that private or confidential information is not made available or disclosed to unauthorized individuals.

Privacy Assures that individuals control how their information is stored, accessed and shared.

Integrity It has two subcategory:

Data integrity Assures that information and programs are changed only in a specified and authorized manner.

System integrity Assures that a system perform its intended function in an unimpaired manner.

Availability Ensuring timely and reliable access to and use of information.

Authenticity & Accountability :

Authenticity The property of being genuine and being able to be verified and trusted. It regards the validity of a trasmission, a message, or message originator. This means verifying that users are who they say they are and that each input arriving at the systme came from a trusted source.

Accountability Ability to uniquely trace the actions of an entity, this supports nonrepudiation, deterrence, fault isolation, intrusion detection and prevention, and after action recovery and legal action.

1.1.0.2 Levels of Impact

Low The loss could be expected to have a limited adverse effect on organizational operations, organizational assets, or individuals

Moderate The loss could be expected to have a serious adverse effect on organizational operations, organizational assets, or individuals

High The loss could be expected to have a severe or catastrophic adverse effect on organizational operations, organizational assets, or individuals

1.1.0.3 Question & Examples

Question *GIVE EXAMPLES OF ASSETS OF LOW, MODERATE AND HIGH IMPACT FOR EACH SECURITY CONCEPT*

Answer

confidentiality :

- **Low Impact**
  - Email addresses of a newsletter mailing list: Unauthorized disclosure of these email addresses might lead to spam but typically has limited consequences.
- **Moderate Impact**
  - Internal company memos: If disclosed, these might reveal business strategies or non-sensitive internal operations, leading to moderate damage.
- **High Impact**
  - Patient medical records: Unauthorized access could lead to severe privacy violations, legal consequences, and loss of trust in the healthcare provider.

integrity :

- **Low Impact**
  - Public website content: Minor, unauthorized changes might not significantly harm the organization but could affect its public image slightly.
- **Moderate Impact**
  - Employee payroll records: Unauthorized alterations could lead to incorrect payments, financial loss, and significant administrative burden to correct.
- **High Impact**
  - Financial transaction records of a bank: Any unauthorized changes could result in severe financial losses, regulatory penalties, and loss of customer trust.

availability :

- **Low Impact**
  - Archived digital marketing materials: Temporary unavailability might cause minor inconvenience but generally won't disrupt business operations.
- **Moderate Impact**
  - Corporate email system: If unavailable, it could disrupt daily operations, delay communications, and affect productivity moderately.
- **High Impact**
  - Core banking systems: Downtime could halt financial transactions, leading to significant financial loss, customer dissatisfaction, and potential regulatory issues.

1.2 Computer Security Challenges

0. Computer Security is no as simple as it might first appear to the novice
1. In developing a particular security mechanism or algorithm, one must always consider potential attacks on those security features
2. Procedures used to provide particular services are often counterintuitive
3. Physical and logical placement of security mechanisms needs to be determined
4. Security mechanism typically based on some secret information: raises question about its creation, distribution, and protection.
5. Attackers only need to find a single weakness, while the designer must find and eliminate all weaknesses to achieve perfect security
6. Security often added as an extra feature once the design is complete ... need for the integration of security in the design processed
7. Security requires regular and constant monitoring
8. There is a natural tendency on the part of users and system managers to perceive little benefit from security investment until a security failure occurs
9. Many users and even security administrators view strong security as an impediment to efficient and user-friendly operation of an information system or use of information

1.2.0.1 Assets of a Computer System

- hardware
- software
- Data
- communication facilities and networks
- Even human resources and physical infrastructures

1.2.0.2 Computer Security Terminolgy

RFC 22828, Internet Security Glossary, May 2000 :

System Resource (Asset) A major application, general support system, high impact program, physical plant, mission critical system, personnel, equipment, or a logically related group of systems.

Adversary (threat agent) individual, group, organization, or government that conducts or has the intent to conduct detrimental activities.

Threat Any circumstance or event with the potential to adversely impact through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service

- organizational operations (including mission, functions, image, or reputation)
- organizational assets
- individuals
- other organizations
- or the Nation itself

Risk A measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of

1. The adverse impacts that would arise if the circumstance or event occurs;
2. the likelihood of occurrence.

Attack Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself

Vulnerability Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source.

Security Policy A set of criteria for the provision of security services. It defines and constrains the activities of a data processing facility in order to maintain a condition of security for systems and data.

Countermeasure A device or techniques that has as its objective :

- the impairment of the operational effectiveness of undesirable or adversarial activity.
- the prevention of espionage, sabotage, theft, or unauthorized access to or use of sensitive information or information systems

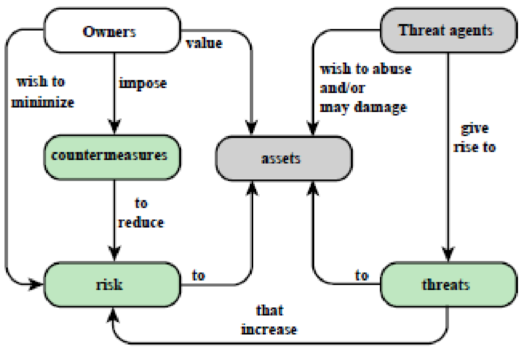


Figure 1: Security Concepts and Relationships

1.2.0.3 Vulnerabilities, Threats and Attacks

Categories of Vulnerabilities the system may become ...

- Corrupted (loss of integrity)
- Leaky (loss of confidentiality)
- unavailable or very slow (loss of availability)

Threats
- Capable of exploiting Vulnerabilities
- Represent potential security harm to an asset

Attacks
- Passive - attempt to learn or make use of information from the system that does not affect system resources
- Active - attempt to alter system resources or affect their operation
- Insider - initiated by an entity inside the security perimeter
- Outsider - initiated from outside the perimeter

1.2.0.4 Countermeasure

Means used to deal with security attacks : **Prevent, Detect, Recover**

May remain residual vulnerabilities

May itself introduce new vulnerabilities

Goal is to minimize residual level of risk to the assets

1.2.0.5 Question & Examples

Question *CONSIDER AN ORGANIZATION MANAGING PUBLIC INFORMATION ON ITS WEB SERVER. ASSIGN A LOW, MODERATE, OR HIGH IMPACT LEVEL FOR THE LOSS OF CONFIDENTIALITY, AVAILABILITY, AND INTEGRITY, RESPECTIVELY. JUSTIFY YOUR ANSWER.*

Answer

Confidentiality: Low Impact
- **Justification:** The information hosted on a public web server is intended for public access. Therefore, the loss of confidentiality would typically not result in significant harm to the organization since the information is already publicly accessible. Unauthorized access to public information does not compromise any private or sensitive data.

Integrity: Moderate Impact
- **Justification:** Maintaining the accuracy and reliability of public information is important to ensure that users receive correct and trustworthy information. Unauthorized changes to the information could mislead the public, harm the organization's reputation, and potentially result in legal consequences if incorrect information causes significant harm. However, the impact might not be as severe as a high-integrity asset like financial records, hence it is considered moderate.

Availability: High Impact
- **Justification:** The availability of the web server is crucial for the organization's operation, especially if the server provides critical services or information to the public. Downtime can lead to loss of service, damage to reputation, and inconvenience to users relying on the information or services provided by the web server. In severe cases, it could also lead to financial losses and a decrease in public trust.

1.3 Threats, Attack and Assets

1.3.1 Threat Consequences

1.3.1.1 Unauthorized disclosure

- It is a threat to **Confidentiality**
- Occurs when an unauthorized entity gains access to data

Attacks

Exposure Sesitive data are directly released to an unauthorized entity

Interception An unauthorized entity directly accesses sensitive data traveling between authorized sources and destinations

Inference An unauthorized entity indirectly accesses sensitive data (but not necessarily the data contained in the communication) by reasoning from characteristics or by-products of communications.

Intrusion An unauthorized entity gains access to sensitive data by circumventing a system's security protections.

1.3.1.2 Deception

- It is a threat to either **system integrity** or **data integrity**
- Occurs when an authorized entity receives false data and believe it is true

Attacks

Masquerade An unauthorized entity gains access to a system or performs a malicious act by posing as an authorized entity.

Falsification False data deceive an authorized entity.

Repudiation An entity deceives another by falsely denying responsibility for an act.

1.3.1.3 Disruption

- It is a threat to **availability** or **system integrity**
- Occurs when the correct system functionality is interrupted or prevented

Attacks

Incapacitation Prevents or interrupts system operation by disabling a system component.

Corruption Undesirably alters system operation by adversely modifying system functions or data.

Obstruction A threat action that interrupts delivery of system services by hindering system operation.

1.3.1.4 Usurpation

- It is a threat to **system integrity**
- Occurs when an unauthorized entity gains control of system services or functions

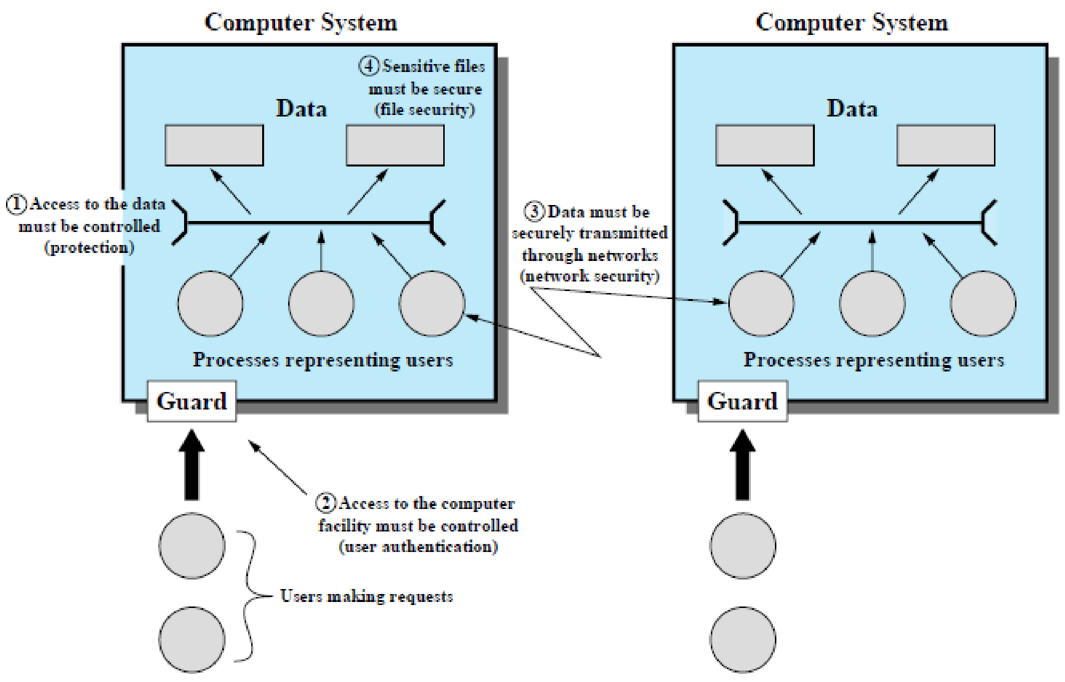
Attacks

Misappropriation An entity assumes unauthorized logical or physical control of a system resource.

Misuse Causes a system component to perform a function or service that is detrimental to system security.

1.3.2 Computer and Network Assets and Threats

	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled thus denying service.	An unencrypted USB drive is stolen.	
Data	Files are encrypted or deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause it to do some unintended task.
Software	Programs are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified, or new files are fabricated
Communication lines and networks	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.



Computer System
Data
④Sensitive files must be secure (file security)
①Access to the data must be controlled (protection)
③Data must be securely transmitted through networks (network security)
Processes representing users
Guard
②Access to the computer facility must be controlled (user authentication)
Users making requests
Computer System
Data
Processes representing users
Guard

Figure 2: Scope of computer security [1]

[1]This figure depicts security concerns other than physical security, including controlling of access to computers systems, safeguarding of data transmitted over communications systems, and safeguarding of stored data

1.3.3 Passive and Active attacks

1.3.3.1 Passive attacks

- Attempts to learn or make use of information from the system but does not affect system resources
- Eavesdropping on, or monitoring of, transmissions
- Goal of attacker is to obtain information that is being transmitted
- Two types:
  - Release of message contents
  - Traffic analysis

1.3.3.2 Active Attacks

- Attempts to alter system resources or affect their operation
- Involve some modification of the data stream or the creation of a false stream
- Four categories:
  - Replay
  - Masquerade
  - Modification of messages
  - Denial of service

1.3.4 Question & Examples

Question *HOW WOULD YOU CLASSIFY THE FOLLOWING ATTACK? UPON STEALING A BANK CUSTOMER'S COMMERCIAL IDENTITY (E.G., THEIR CREDIT CARD OR ACCOUNT INFORMATION), THE ATTACKER PRESENTS THOSE CREDENTIALS FOR THE MALICIOUS PURPOSE OF USING THE CUSTOMER'S CREDIT LINE TO STEAL MONEY*

1.4 Functional requirements

1.4.1 Classification of Contermeasures

Several classifications of countermeasures aimed at reducing vulnerabilities and dealing with threats. Classification according to functional requirements: defined in FIPS 200 (Minimum Security Requirements for Federal Informationand Information Systems). This standard enumerates 17 security-related areas: They encompass a wide range of countermeasures to security vulnerabilities and threats. Roughly, they can be divided into two categories: **technical measures** and **management issues**.

1.4.1.1 Technical Measures

Access Control **Limit** information system access to authorized users and of processes that act on their behalf; **Limit** the types of transactions and functions that authorized users are permitted to exercise.

Identification and Authentication **Identify** system users, processes acting on behalf of users, or devices; **Authenticate** (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access.

System and Communications Protection :

- Monitor, control, and protect organizational communications (both transmitted and received) at the external boundaries and key internal boundaries of the information systems
- Employ architectural designs, software development techniques, and systems engineering principles that promote effective information security within organizational information systems

System and Information Integrity :

- Identify, report, and correct information and information system flaws in a timely manner
- Provide protection from malicious code at appropriate locations within organizational information systems
- Monitor information system security alerts and advisories and take appropriate actions in response

1.4.1.2 Management measures

Systems and Services Acquisition :

- Allocate sufficient resources to adequately protect organizational information systems
- Employ system development life cycle processes that incorporate information security considerations
- Employ software usage and installation restrictions
- Ensure that third-party providers employ adequate security measures to protect information, applications, and/or services outsourced from the organization

Awareness and Training :

- Address managers and users of organizational information systems
- Make them aware of the security risks and of the applicable laws, regulations, and policies
- Train all the personnel to carry out their duties according to the security protocols

Audit and Accountability :Create, protect, and retain information system audit records to enable the monitoring, analysis, investigation, and reporting of unlawful, unauthorized, or inappropriate information system activity. Ensure that the actions of individual users can be uniquely traced so they can be held accountable for their actions.

Certification, Accreditation, and Security Assessments :

- Periodically assess the security controls to determine their effectiveness
- Develop and implement plans of action designed to correct deficiencies and reduce or eliminate vulnerabilities
- Monitor information system security controls on an ongoing basis to ensure the continued effectiveness of the controls

Contingency Planning :

- Establish, maintain, and implement plans for emergency response, backup operations, and post-disaster recovery

- Ensure the availability of critical information resources and continuity of operations in emergency situations

Maintenance :

- Perform periodic and timely maintenance on organizational information systems
- Provide effective controls on the tools, techniques, mechanisms, and personnel used to conduct information system maintenance

Physical and Environmental Protection :

- Limit physical access to information systems, equipment, and the respective operating environments to authorized individuals
- Protect the physical plant and support infrastructure for information systems
- Provide supporting utilities for information systems
- Protect information systems against environmental hazards
- Provide appropriate environmental controls in facilities containing information systems

Planning :

- Develop, document, periodically update, and implement security plans
- The security plans describe the security controls in place or planned for the information systems and the rules of behavior for individuals accessing the system

Personnel Security :

- Ensure that individuals with positions of responsibility (including third-parties) are trustworthy and meet the security criteria for those positions
- Ensure that organizational information and information systems are protected during and after personnel actions such as terminations and transfers
- Employ formal sanctions for personnel failing to comply with organizational security policies and procedures

Risk Assessment :

- Periodically assess the risk to organizational operations (including mission, functions, image, or reputation), organizational assets, and individuals
- Risks may result from the operation of organizational information systems and the associated processing, storage, or transmission of organizational information

1.4.1.3 Overlap functional areas

Configuration Management :

- Establish and maintain baseline configurations and inventories of organizational information systems (hardware, software, documents, etc.) throughout the respective system development life cycles
- Establish and enforce security configuration settings for information technology products employed

Incident Response :

- Establish an operational incident-handling capability (includes preparation, detection, analysis, containment, recovery, and user-response activities)
- Track, document, and report incidents to appropriate organizational officials and/or authorities

Media Protection :

- Protect information system media, both paper and digital
- Limit access to information on information system media to authorized users
- Sanitize or destroy information system media before disposal or release for reuse

1.4.2 Fundamental Security Design Principles

Economy of Mechanism Keep security measures simple and small to reduce vulnerabilities and maintenance efforts.

Fail-safe Default Base access decisions on permission rather than exclusion to ensure a safer failure mode.

Complete Mediation Check every access against the access control mechanism to prevent reliance on cached decisions.

Open Design Design security mechanisms openly to allow for public scrutiny and confidence in their effectiveness.

Separation of Privilege Use multiple privilege attributes to access restricted resources, mitigating potential damage from security attacks.

Least Privilege Restrict processes and users to the minimum set of privileges required for their tasks to minimize risks.

Least Common Mechanism Minimize shared functions between users to reduce unintended communication paths and security implications.

Psychological Acceptability Ensure security mechanisms do not unduly interfere with user work and are transparent and intuitive.

Isolation Isolate public access systems from critical resources and isolate processes and files of individual users to prevent unauthorized access.

Encapsulation Protect data objects by encapsulating procedures and data objects within their own domain.

Modularity Develop security functions as separate modules and use a modular architecture to support upgrades without redesigning the entire system.

Layering Employ multiple, overlapping protection approaches to address people, technology, and operational aspects of information systems.

Least Astonishment Ensure programs and user interfaces respond in a way that is least likely to astonish users, promoting intuitive understanding of security mechanisms.

1.4.3 Question & Examples

Question :

1. Encryption of the SSD of a laptop
2. Keeping the hash of executable files of the OS in a CD
3. Uninstallation from the system of unused applications

Answer :

1. Media Protection
2. System and Information integrity
3. Configuration Management

Question :
Consider the following general code for allowing access to a resource:

```
DWORD dwRet= IsAccessAllowed(...);


if (dwRet== ERROR_ACCESS_DENIED) {
    // Security check failed.
    // Inform user that access is denied.
} else {
    // Security check OK.
}
```

- Explain the security flaw in this program.
- Rewrite the code to avoid the flaw.

Answer Chiedere a lezione ...

1.5 Attack surfaces and security strategies

1.5.1 Attack surfaces

Consist of the reachable and exploitable vulnerabilities in a system Examples:

- Open ports on outward facing Web and other servers, and code listening on those ports
- Services available on the inside of a firewall
- Code that processes incoming data, email, XML, office documents, and industry-specific custom data exchange formats
- Interfaces, SQL, and Web forms
- An employee with access to sensitive information vulnerable to a social engineering attack

1.5.1.1 Categories

Network Attack Surface

- Vulnerabilities over an enterprise network, wide-area network, or the Internet
- Network protocol vulnerabilities (e.g., those used for DoS attacks)
- Disruption of communications links
- Various forms of intruder attacks

Software Attack Surface

- Vulnerabilities in application, utility, or operating system code
- Particular focus on Web server software

Human Attack Surface

- Vulnerabilities created by personnel or outsiders, such as social engineering, Human error, Trusted insiders

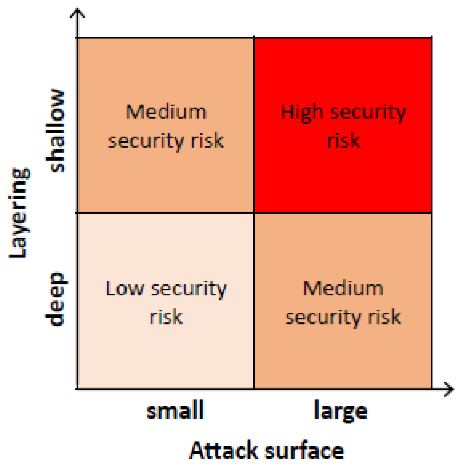
1.5.1.2 Analysis

To assess the scale and severity of threats to a system:

- Identifies the points of vulnerability
- Identifies the most appropriate security mechanisms and their deployment

Uses:

- Makes developers and security analysts aware of where security mechanisms are required.
- Designers may be able to find ways to make the surface smaller, thus making the task of the adversary more difficult.
- It also provides guidance on setting priorities for testing, strengthening security measures, or modifying the service or application.



1.5.1.3 Trees

An attack tree is a branching, hierarchical data structure that represents a set of potential techniques for exploiting security vulnerabilities:

- The root of the tree is the security incident (the goal of the attack).
- The branches and subnodes of the tree are the ways that an attacker could reach that goal.
- Each subnode defines a subgoal, and each subgoal may have its own set of further subgoals, etc.
- The leaf nodes represent different ways to initiate an attack.

Each node other than a leaf is either an AND-node or an OR-node:

- To achieve the goal represented by an AND-node, the subgoals represented by all of that node's subnodes must be achieved.
- For an OR-node, at least one of the subgoals must be achieved.

Branches can be labeled with values representing difficulty, cost, or other attack attributes, so that alternative attacks can be compared.

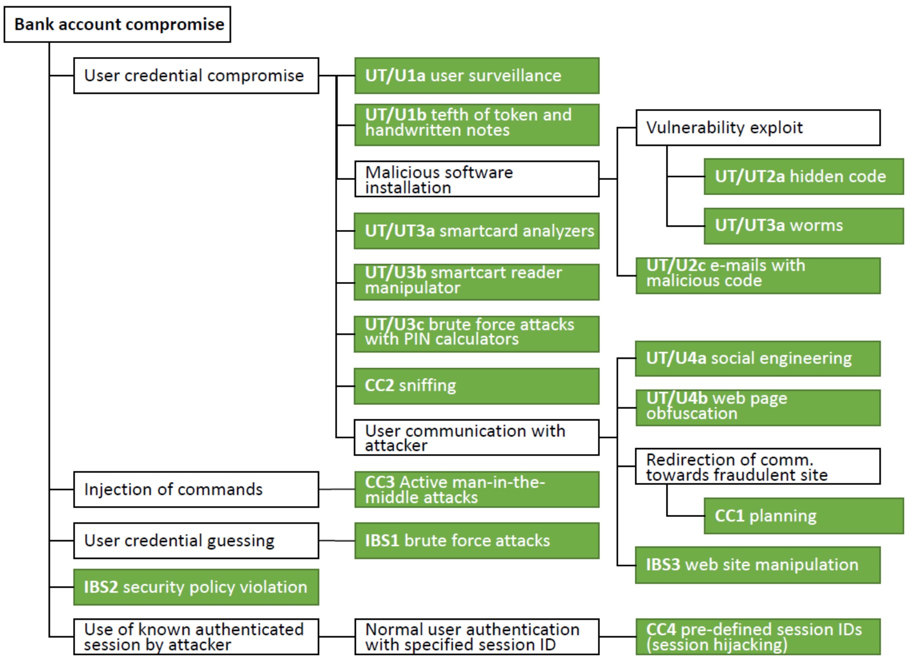

Bank account compromise
User credential compromise
UT/U1a user surveillance
UT/U1b tefth of token and handwritten notes
Malicious software installation
Vulnerability exploit
UT/UT2a hidden code
UT/UT3a worms
UT/U2c e-mails with malicious code
UT/UT3a smartcard analyzers
UT/U3b smartcart reader manipulator
UT/U3c brute force attacks with PIN calculators
CC2 sniffing
User communication with attacker
UT/U4a social engineering
UT/U4b web page obfuscation
Redirection of comm. towards fraudulent site
CC1 planning
IBS3 web site manipulation
Injection of commands
CC3 Active man-in-the-middle attacks
User credential guessing
IBS1 brute force attacks
IBS2 security policy violation
Use of known authenticated session by attacker
Normal user authentication with specified session ID
CC4 pre-defined session IDs (session hijacking)

Figure 3: An attack tree for internet banking authentication

Attack trees allow analysis of attack patterns:

- To build a body of knowledge about both attack strategies and patterns.
- To document security attacks in a structured form that reveals key vulnerabilities.
- To guide the design of systems and applications, and the choice and strength of countermeasures.

1.5.1.4 Question & Examples

Q: CAN YOU MAKE AN EXAMPLE OF SOMETHING THAT MAY ENLARGE THE ATTACK SURFACE ON YOUR PC? **A:** Example of Enlarging the Attack Surface on Your PC One example of something that may enlarge the attack surface on your PC is installing unnecessary software or applications. Each piece of software you install can introduce new vulnerabilities that attackers might exploit. Here is a detailed breakdown of how this can happen:

Increased Number of Potential Vulnerabilities

- Each installed application has its own codebase and, potentially, its own vulnerabilities. By adding more software, you increase the total number of vulnerabilities that might exist on your system.
- **Example:** Installing a new media player that has a known buffer overflow vulnerability.

Additional Network Services

- Some software might install additional network services or open network ports, which could be exploited by attackers.
- **Example:** A file-sharing application that opens new ports and can be accessed remotely.

Expanded Attack Vectors

- New software can introduce new attack vectors. For instance, a program that integrates with your browser might create opportunities for drive-by download attacks or malicious extensions.
- **Example:** Installing a browser toolbar that inadvertently allows cross-site scripting (XSS) attacks.

Third-Party Software Dependencies

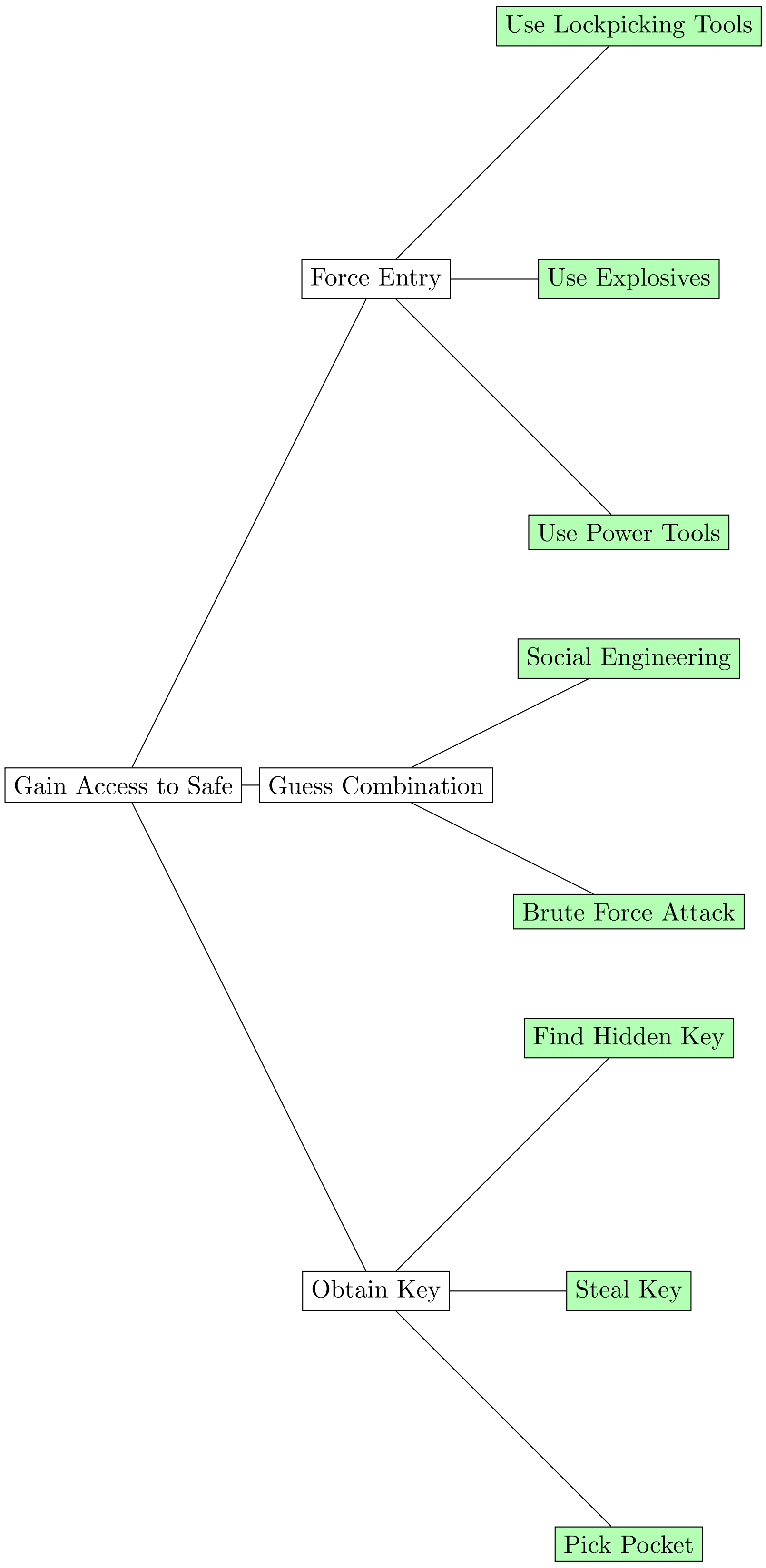
- Many applications rely on third-party libraries or components. These dependencies might not be updated as frequently and could contain vulnerabilities.
- **Example:** An outdated version of a library used by a photo editing software.

Increased Complexity

- The more software you have, the more complex your system becomes. This increased complexity can make it harder to manage and secure your system effectively.
- **Example:** Multiple security tools installed that may conflict with each other, reducing overall security effectiveness.

Q: DRAW AN ATTACK TREE FOR GAINING ACCESS TO THE CONTENT OF A PHYSICAL SAFE

A:



1.5.2 Security strategies

1.5.2.1 Security Policy

Several factors to be considered in the development of a security policy:

- The value of the assets being protected
- The vulnerabilities of the system
- Potential threats and the likelihood of attacks

Relevant tradeoffs:

- Ease of use versus security
- Cost of security versus cost of failure and recovery

The security policy is a business decision, possibly influenced by legal requirements.

1.5.2.2 Security implementation

Prevention: An ideal security scheme is one in which no attack is successful…

- Example: transmission of encrypted data

Detection: Absolute protection is often not feasible, but it is practical to detect security attacks.

- Example: intrusion detection systems designed to detect the presence of unauthorized individuals logged onto a system.

Response: If security mechanisms detect an ongoing attack, the system may be able to respond in such a way as to halt the attack and prevent further damage.

Recovery: An example of recovery is the use of backup systems, so that if data integrity is compromised, a prior, correct copy of the data can be reloaded.

1.5.2.3 Assurance and Evaluation

Assurance is an attribute of a system that provides confidence that the system operates such that the system's security policy is enforced.

- Encompasses both system design and implementation.
- Deals with the questions:
  - "Does the security system design meet its requirements?"
  - "Does the security system implementation meet its specifications?"
- It is expressed as a degree of confidence, not in terms of a formal proof that a design or implementation is correct.
- Absolute proofs not possible at the state of the art.

Evaluation is the process of examining a computer product or system with respect to certain criteria.

- Involves testing, formal analytic, or mathematical techniques.

1.6 Standards

National Institute of Standards and Technology (NIST): NIST is a U.S. federal agency that deals with measurement science, standards, and technology related to U.S. government use and to the promotion of U.S. private sector innovation.

Internet Society (ISOC): ISOC is a professional membership society that provides leadership in addressing issues that confront the future of the Internet, and is the organization home for the groups responsible for Internet infrastructure standards.

International Telecommunication Union (ITU-T): ITU is a United Nations agency in which governments and the private sector coordinate global telecom networks and services.

International Organization for Standardization (ISO): ISO is a nongovernmental organization whose work results in international agreements that are published as International Standards.

1.7 Questions & Examples

Question CONSIDER AN INFORMATION SYSTEM OF A SMART CITY IN WHICH CITIZENS PROVIDE AN ID NUMBER AND A E-DOCUMENT FOR ACCOUNT ACCESS THROUGH AUTOMATIC MACHINES. GIVE EXAMPLES OF CONFIDENTIALITY, INTEGRITY, AND AVAILABILITY REQUIREMENTS ASSOCIATED WITH THE SYSTEM. IN EACH CASE, INDICATE THE DEGREE OF THE IMPORTANCE OF THE REQUIREMENT.

Answer

- **Confidentiality Requirements:**
  - The ID number and e-document provided by citizens should be encrypted during transmission and storage to prevent unauthorized access.
  - Importance : High. Ensuring confidentiality is crucial to protect citizens' sensitive personal information from unauthorized disclosure or misuse.
- **Integrity Requirements:**
  - The information system should have mechanisms to detect unauthorized modification or tampering with citizens' ID numbers or e-documents.
  - Importance: High. Maintaining the integrity of the data is essential to ensure that citizens can trust the accuracy and reliability of the information provided by the system.
- **Availability Requirements:**
  - The automatic machines should be operational and accessible to citizens 24/7, with minimal downtime for maintenance or repairs.
  - Importance: High. The availability of the system is critical to ensure that citizens can access the services they need at any time, especially in emergency situations.

Question CONSIDER A LAW ENFORCEMENT ORGANIZATION MANAGING EXTREMELY SENSITIVE INVESTIGATIVE INFORMATION. ASSIGN A LOW, MODERATE, OR HIGH IMPACT LEVEL FOR THE LOSS OF CONFIDENTIALITY, AVAILABILITY, AND INTEGRITY, RESPECTIVELY. JUSTIFY YOUR ANSWER.

Answer

Confidentiality Impact:
- Loss of confidentiality: High
  - Justification: The loss of confidentiality of sensitive investigative information could compromise ongoing investigations, reveal the identities of informants or undercover agents, and jeopardize national security or public safety.

Availability Impact:
- Loss of availability: Moderate
  - Justification: While the loss of availability could disrupt access to critical information during investigations or operations, law enforcement organizations often have backup systems and procedures in place to mitigate the impact of temporary downtime.

Integrity Impact:
- Loss of integrity: High
  - Justification: The loss of integrity could result in the manipulation or alteration of investigative data, leading to false evidence, wrongful convictions, or the compromise of the integrity of legal proceedings, undermining the trust in the justice system.

2 Cryptogrphic tools

2.1 Symmetric Encryption

The universal technique for providing confidentiality for transmitted or stored data also referred to as conventional encryption or single-key encryption. Two requirements for secure use:

- Need a strong encryption algorithm
- Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure

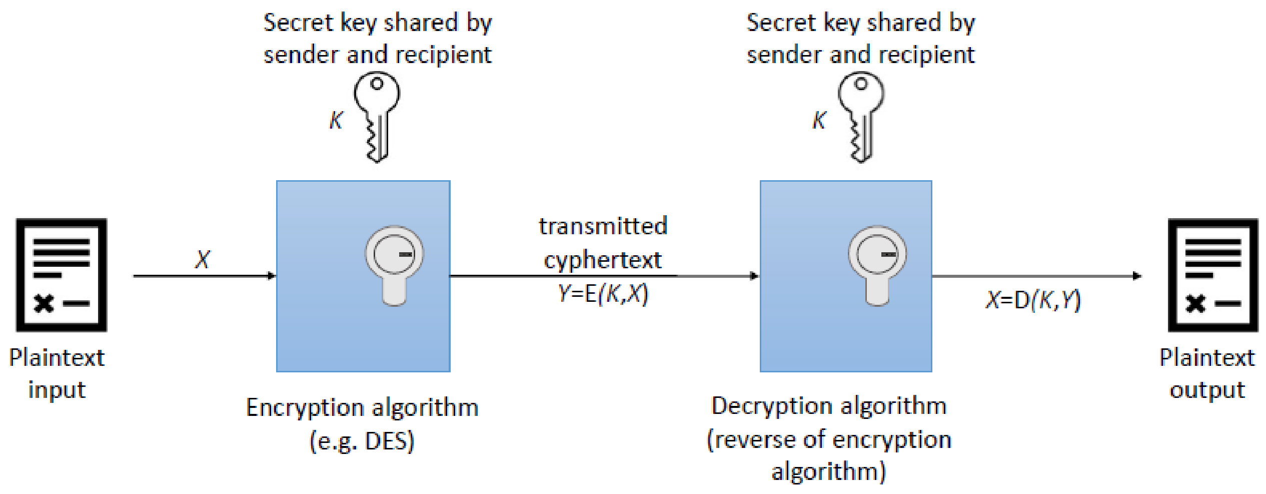

Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
Plaintext input
X
transmitted cyphertext
Y=E(K,X)
X=D(K,Y)
Plaintext output
Encryption algorithm (e.g. DES)
Decryption algorithm (reverse of encryption algorithm)

Figure 4: Simplified model of symmetric Encryption

2.1.1 Attacking symmetric encryption

2.1.1.1 Cryptoanalytic attacks

- Rely on:
  - Nature of the algorithm
  - Some knowledge of the general characteristics of the plaintext
  - Some sample plaintext-ciphertext pairs
- Exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or the key being used
  - If successful, all future and past messages encrypted with that key are compromised

2.1.1.2 Brute force attacks

- Try all possible keys on some ciphertext until an intelligible translation into plaintext is obtained
  - On average half of all possible keys must be tried to achieve success

2.1.2 Algorithm

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128,192, or 256

Table 1: Comparison of Three Popular Symmetric Encryption Algorithms

2.1.2.1 DES: Data Encryption Standard

- Until recently was the most widely used encryption scheme
  - FIPS PUB 46 (January 1977)
  - Referred to as the Data Encryption Algorithm (DEA)
  - Uses 64 bit plaintext block and 56 bit key to produce a 64 bit ciphertext block
- Strength concerns:
  - Concerns about the algorithm itself
    * DES is the most studied encryption algorithm in existence
  - Concerns about the use of a 56-bit key
    * The speed of commercial off-the-shelf processors makes this key length woefully inadequate

2.1.2.2 Triple DES: 3DES

- Repeats basic DES algorithm three times using either two or three unique keys
- First standardized for use in financial applications in ANSI standard X9.17 in 1985
- Attractions:
  - 168-bit key length overcomes the vulnerability to brute-force attack of DES
  - Underlying encryption algorithm is the same as in DES
- Drawbacks:
  - Algorithm is sluggish in software
  - Uses a 64-bit block size

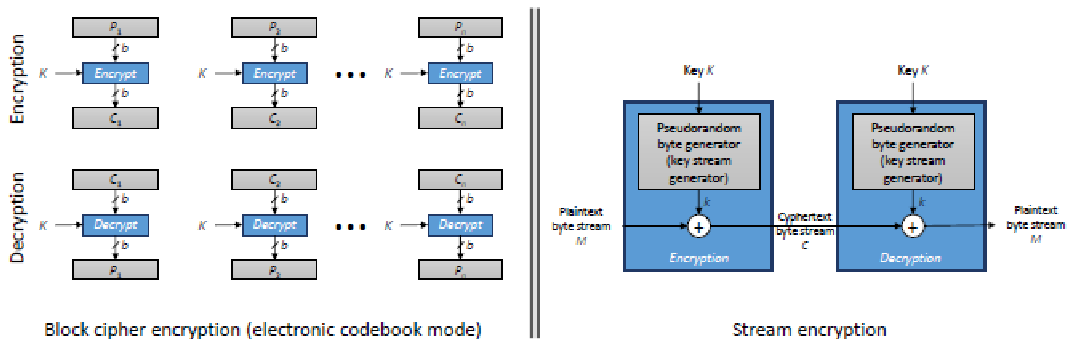
2.1.2.3 AES: Advanced Encryption Standard

- Needed a replacement for 3DES
  - Should have a security strength equal to or better than 3DES
  - Significantly improved efficiency
  - Symmetric block cipher
  - 128-bit data and 128/192/256-bit keys
- NIST called for proposals for a new AES in 1997
  - Published as FIPS 197
  - Selected Rijndael in November

2.1.2.4 Pratical Security Issues

- Typically symmetric encryption is applied to a unit of data larger than a single 64-bit or 128-bit block
- Electronic codebook (ECB) mode is the simplest approach to multiple-block encryption
  - Each block of plaintext is encrypted using the same key
  - Cryptanalysts may be able to exploit regularities in the plaintext
- Modes of operation
  - Alternative techniques developed to increase the security of symmetric block encryption for large sequences
  - Overcomes the weaknesses of ECB

2.1.3 Types of symmetric encryption



2.1.3.1 Block cyphers

- With the Electronic Codebook (ECB) mode:
  - A plaintext of length nb is divided into n b-bit blocks (P1, P2, ..., Pn).
  - Each block is encrypted using the same algorithm and the same encryption key, to produce a sequence of n b-bit blocks of ciphertext (C1, C2, ..., Cn).
- Security concerns:
  - A cryptanalyst may exploit regularities in the plaintext to ease the task of decryption.
  - For example, if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext-ciphertext pairs to work with.

2.1.3.2 Stream cyphers

- The key is input to a pseudorandom bit generator that produces a stream of numbers that are apparently random.
  - A pseudorandom stream is unpredictable without knowledge of the input key.
- The output of the generator, called a keystream, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation.
- A stream cipher can also operate on one bit at a time or on units larger than a byte at a time.
- A stream cipher can be as secure as a block cipher of comparable key length.
  - With a properly designed pseudorandom number generator.
- Stream ciphers are typically faster and use far less code than block ciphers.
- However, with a block cipher the keys can be reused.

- Stream ciphers are good for encryption/decryption of data streams.
  - Data communications channel or a browser/Web link.
- Block ciphers are good for file encryption, e-mail, databases etc.
- However, both can be used in virtually any application.

Block Cipher	**Stream Cipher**
• Processes the input one block of elements at a time • Produces an output block for each input block • Can reuse keys • More common	• Processes the input elements continuously • Produces output one element at a time • Primary advantage is that they are almost always faster and use far less code • Encrypts plaintext one byte at a time • Pseudorandom stream is one that is unpredictable without knowledge of the input key

Table 2: Difference

2.1.4 Questions & Examples

Question HOW IS CRYPTANALYSIS DIFFERENT FROM BRUTEFORCE ATTACK?

Answer

- **Approach**:
  - **Cryptanalysis**: Involves understanding and exploiting weaknesses in the encryption algorithm or its implementation.
  - **Brute-Force Attacks**: Involve trying every possible key to decrypt the ciphertext.
- **Knowledge Requirement**:
  - **Cryptanalysis**: Often requires specialized knowledge of mathematics, statistics, or the specific encryption algorithm.
  - **Brute-Force Attacks**: Do not require specific knowledge beyond the ability to generate keys and test them.
- **Speed**:
  - **Cryptanalysis**: Can potentially be faster than brute-force attacks if effective vulnerabilities are identified, especially for large key sizes where brute-force becomes impractical.

Question WHAT ARE THE (TWO) PRINCIPAL REQUIREMENTS FOR THE SECURE USE OF SYMMETRIC ENCRYPTION?

Answer

1. **Key Management**:
   - Secure generation of keys: Keys should be generated using a secure random number generator.
   - Distribution of keys: Keys must be securely distributed to authorized parties without interception by unauthorized entities.
   - Key storage: Keys should be stored securely to prevent unauthorized access.
   - Key update and rotation: Regular updates of keys and periodic rotation are essential to mitigate the risk of compromise.
2. **Algorithm Selection and Implementation**:
   - Choosing strong algorithms: Use of well-established and cryptographically strong algorithms (e.g., AES-256) that are resistant to known attacks.
   - Secure implementation: Implementation of the encryption algorithm must follow best practices and standards to avoid vulnerabilities such as side-channel attacks or implementation flaws.
   - Regular security audits: Periodic security audits and evaluations of the encryption implementation to ensure it meets current security standards and practices.

2.2 Message authentication and hash functions

2.2.1 Message authentication

2.2.1.1 Message authentication without confidentiality: Message encryption by itself does not provide a secure form of authentication. It is possible to combine authentication and confidentiality in a single algorithm by encrypting a message along with its authentication tag. Typically, message authentication is provided as a separate function from message encryption. Situations in which message authentication without confidentiality may be preferable include:

1. There are a number of applications in which the same message is broadcast to a number of destinations.
2. An exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages.
3. Authentication of a computer program in plaintext is an attractive service.

Thus, there is a place for both authentication and encryption in meeting security requirements

2.2.1.2 MAC: Message Authentication Code A small block of data (the message authentication code – MAC) is appended to the message to be authenticated MAC generated by means of a secret key:

$$\text{MAC} = \text{Func}(\text{Key}, \text{Message})$$

The secret key is shared between the two communicating parties

Properties

1. **Message Integrity**
   - The receiver is assured that the message has not been altered.
   - If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code.
   - Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.
2. **Message Authentication**
   - The receiver is assured that the message is from the alleged sender.
   - Because no one else knows the secret key, no one else could prepare a message with a proper code.
3. **Proper Sequence Assurance**
   - The receiver is assured of the proper sequence if the message includes a sequence number.
   - As in X.25, HDLC, and TCP.
   - The attacker cannot successfully alter the sequence number.

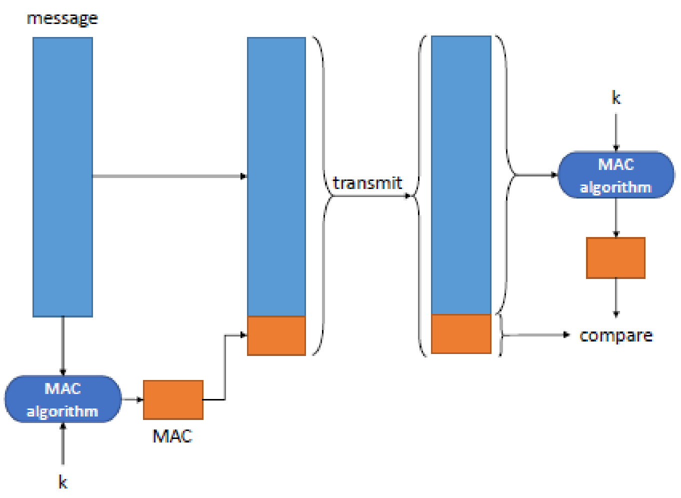
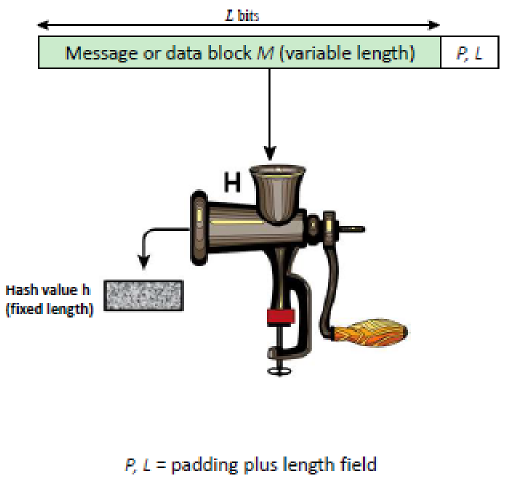


Figure 5: Message authentication using a MAC


L bits
Message or data block M (variable length)
P, L
H
Hash value h (fixed length)
P, L = padding plus length field

2.2.2 One-way hash functions

2.2.2.1 Propreties

1. It can be applied to data blocks of any size.
2. It produces a fixed length output.
3. It is easy to compute (making both software or hardware implementations practical).
4. It is one way: it is computationally infeasible to find x such that $H(x) = h$.
5. It is weak collision resistant: given x it is computationally infeasible to find y such that $H(x) = H(y)$.
6. It is collision resistant: it is computationally infeasible to find a pair x, y such that $H(x) = H(y)$.

2.2.2.2 Message authentication with one-way hash functions The hash function takes a variable-size message M and produces a fixed-size message digest $H(M)$. Typically, the message is padded out to an integer multiple of some fixed length (e.g., 1024 bits). The padding includes the value of the length of the original message in bits. The length field is a security measure to increase the difficulty for an attacker to produce an alternative message with the same hash value. Unlike encryption algorithms, the hash function does not need a secret key.

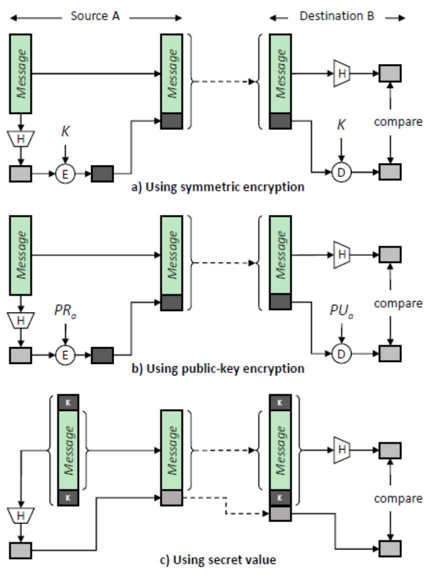


Figure 6: Alternative way

2.2.3 Alternative way of authenticating a message

a The message digest is encrypted using symmetric encryption.

- Only the sender and receiver know the encryption key.
- Assures the authenticity of the digest.

b The message digest is encrypted using public-key encryption.

- Provides a digital signature as well as message authentication.
- Does not require the distribution of keys to communicating parties.

c The method avoids encryption!

- **Advantages:**
  - Software encryption is rather slow (even if the message is short, there may be several messages).
  - Hardware encryption has costs and it is optimized for large data sizes.
  - The encryption algorithm may be patented (and thus subject to royalties).
- **Method based on the keyed hash MAC technique:**
  - Assumes that two communicating parties, say A and B, share a common secret key K.
  - As long as K is secret, there's no way for the attacker to modify the message or to generate a false message.
  - Using K at the beginning and at the end makes the scheme more secure.

2.2.4 Hash function for message authentication

2.2.4.1 Proprieties

- Can be applied to a block of data of any size.
- Produces a fixed-length output.
- $H(x)$ is relatively easy to compute for any given x.
- One-way or pre-image resistant(In case (c), if this was not true it would be possible to find the secret key k!):
  - Given h, it is computationally infeasible to find x such that $H(x) = h$.
- Given x, it is computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$. (Necessary to prevent forgery in cases (a), (b) and (c))
- Collision resistant or strong collision resistance: Computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
  - Protects against an attack in which one party generates a message for another party to sign.
  - For example:
    - Suppose Bob gets to write an "I Owe You" message, sends it to Alice, and she signs it.
    - Bob forges two messages with the same hash:
      - One message requires Alice to pay a small amount.
      - Another message requires Alice to pay a large amount.
    - Alice signs the first message, and Bob is then able to claim that the second message is authentic.

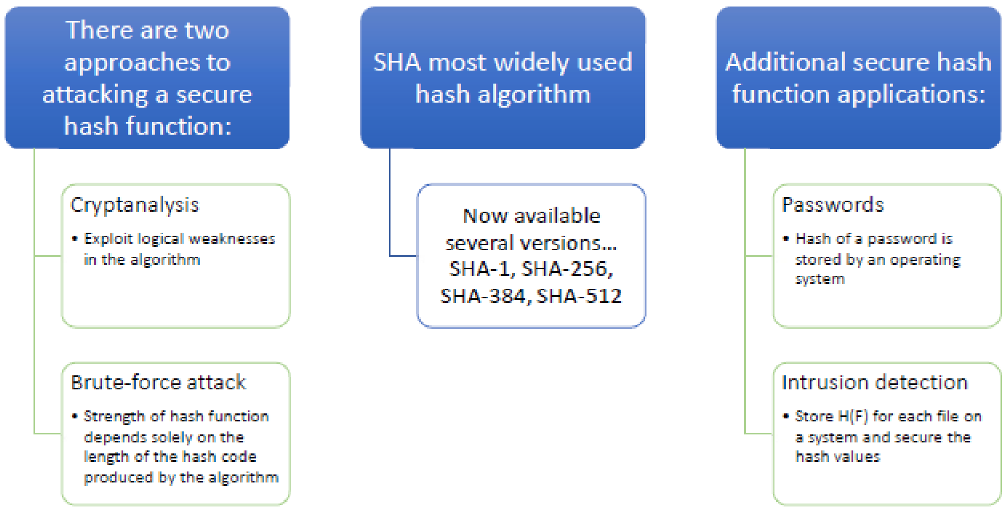
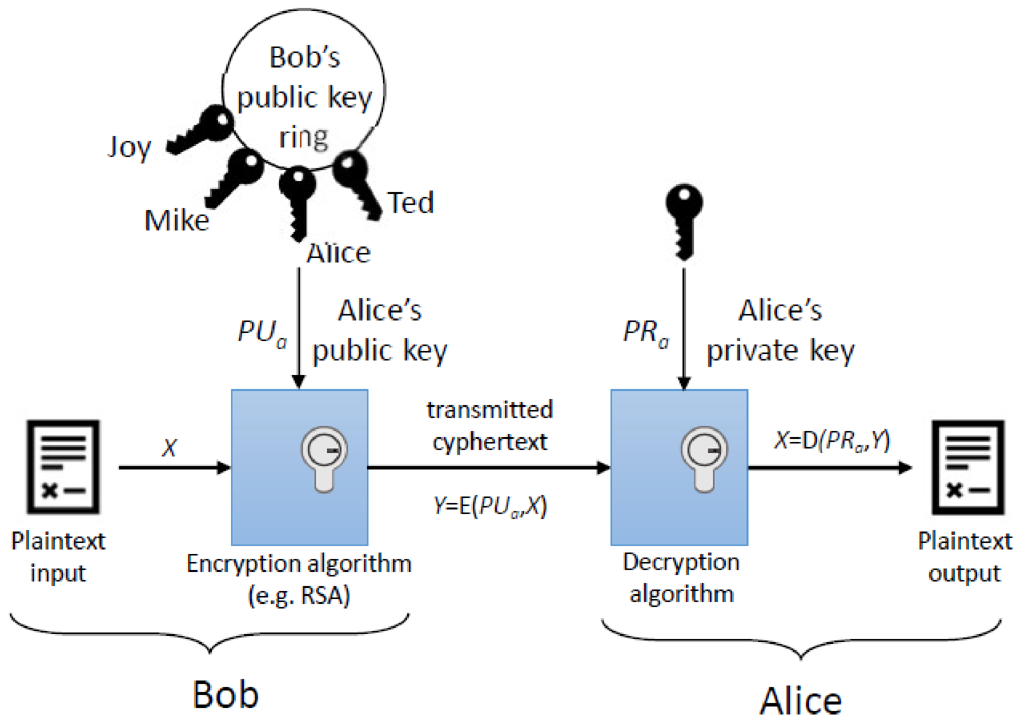

There are two approaches to attacking a secure hash function:
Cryptanalysis
• Exploit logical weaknesses in the algorithm
Brute-force attack
• Strength of hash function depends solely on the length of the hash code produced by the algorithm
SHA most widely used hash algorithm
Now available several versions... SHA-1, SHA-256, SHA-384, SHA-512
Additional secure hash function applications:
Passwords
• Hash of a password is stored by an operating system
Intrusion detection
• Store H(F) for each file on a system and secure the hash values

Figure 7: Security of hash functions

Further applications of secure hash functions

- **Passwords:**
  - Some Operating Systems store hashes rather than passwords.
  - When a user enters a password, the hash of that password is compared to the stored hash value for verification.
  - The actual password is not retrievable even when accessing the password file.
- **Intrusion detection:**
  - Store $H(F)$ for each file on a system and keep the hash values secure (e.g., on a CD-R that is kept secure).
  - One can later determine if a file has been modified by recomputing $H(F)$.
  - An intruder would need to change F without changing $H(F)$, which is not computationally feasible.



2.2.5 Question & Examples

Question WHICH ONE OF THE PREVIOUS SCHEMAS GUARANTEES THE NONREPUDIATION PROPERTY? (About Figure 6)

Answer The schema that guarantees the non-repudiation property is:

1. The message digest is encrypted using public-key encryption.
   - Provides a digital signature as well as message authentication.
   - Does not require the distribution of keys to communicating parties.

This method provides a digital signature, which is crucial for non-repudiation. It ensures that the sender cannot deny having sent the message because the public-key encryption provides a verifiable proof of the sender's identity.

Question WHY DOES THE HASH FUNCTION NEED TO BE ONE-WAY?

Answer The hash function needs to be one-way primarily for security reasons. This property ensures that given the output (hash value), it is computationally infeasible to determine the original input (message or data) that produced that hash.

2.3 Public-key encryption

2.3.1 Public-key encryption scheme

2.3.1.1 Public key

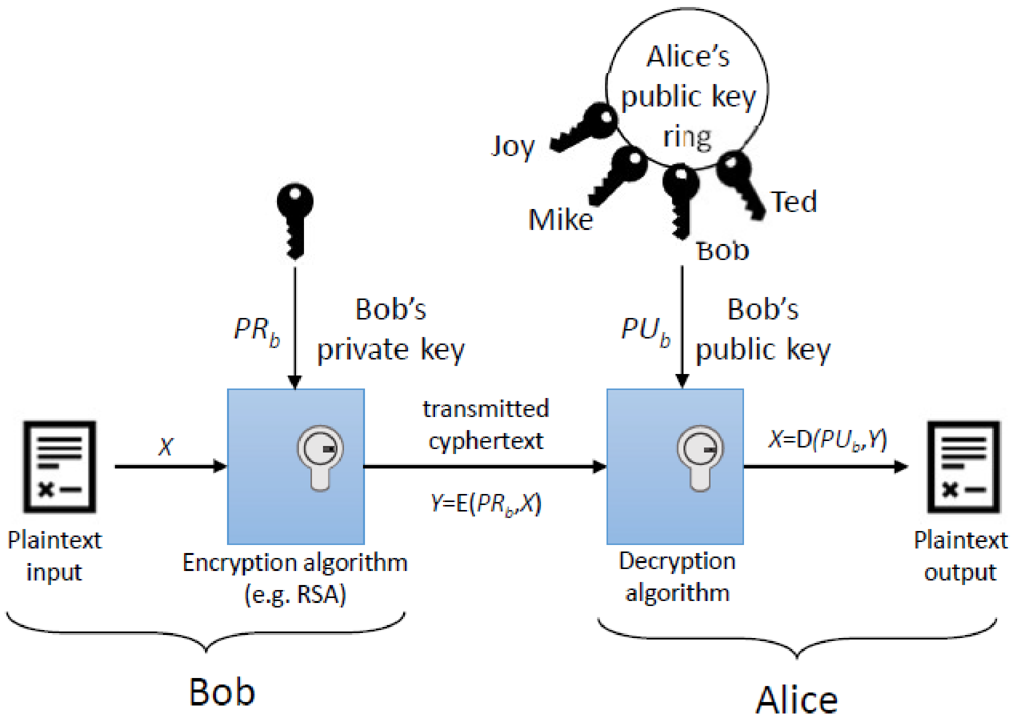
Plaintext Readable message or data that is fed into the algorithm as input

Encryption algorithm Performs transformations on the plaintext

Public and private key Pair of keys, one for encryption, one for decryption

Ciphertext Scrambled message produced as output

Decryption key Produces the original plaintext



2.3.1.2 Private key

- User encrypts data using his or her own private key
- Anyone who knows the corresponding public key will be able to decrypt the message

2.3.2 Public-key Cryptosystems

2.3.2.1 Requirements

- Computationally easy to create key pairs
- Useful if either key can be used for Requirements each role
- Computationally easy for sender knowing public key to encrypt messages
- Computationally infeasible for opponent to determine private key from public key
- Computationally easy for receiver knowing private key to decrypt ciphertext
- Computationally infeasible for opponent to otherwise recover original message

2.3.2.2 Algorithms

- RSA (Rivest, Shamir, Adleman)
  - Developed in 1977
  - Most widely accepted and implemented approach to public-key encryption
  - Block cipher in which the plaintext and ciphertext are integers between 0 and $n-1$ for some n
- Diffie-Hellman key exchange algorithm
  - Enables two users to securely reach agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages
  - Limited to the exchange of the keys
- Digital Signature Standard (DSS)
  - Provides only a digital signature function with SHA-1
  - Cannot be used for encryption or key exchange
- Elliptic curve cryptography (ECC)
  - Security like RSA, but with much smaller keys

Algorithm	**Digital signature**	**Symmetric key distribution**	**Encryption of secret keys**
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

2.3.3 Digital Signatures

NIST FIPS PUB 186-4 defines a digital signature as: "*The result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for verifying origin authentication, data integrity and signatory non-repudiation.*" Thus, a digital signature is a data-dependent bit pattern, generated by an agent as a function of a file, message, or other form of data block FIPS 186-4 specifies the use of one of three digital signature algorithms:

DSA Digital Signature Algorithm

RSA Digital Signature Algorithm

ECDSA Elliptic Curve Digital Signature Algorithm

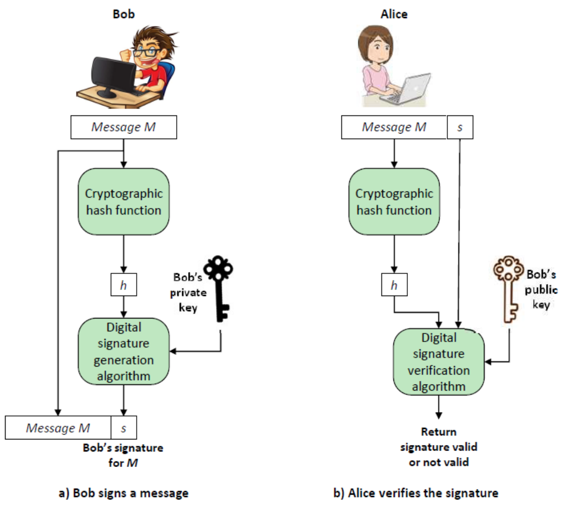

Bob
Message M
Cryptographic hash function
h
Bob's private key
Digital signature generation algorithm
Message M
s
Bob's signature for M
a) Bob signs a message
Alice
Message M
s
Cryptographic hash function
h
Bob's public key
Digital signature verification algorithm
Return signature valid or not valid
b) Alice verifies the signature

Figure 8: Essential elements of a digital signature process

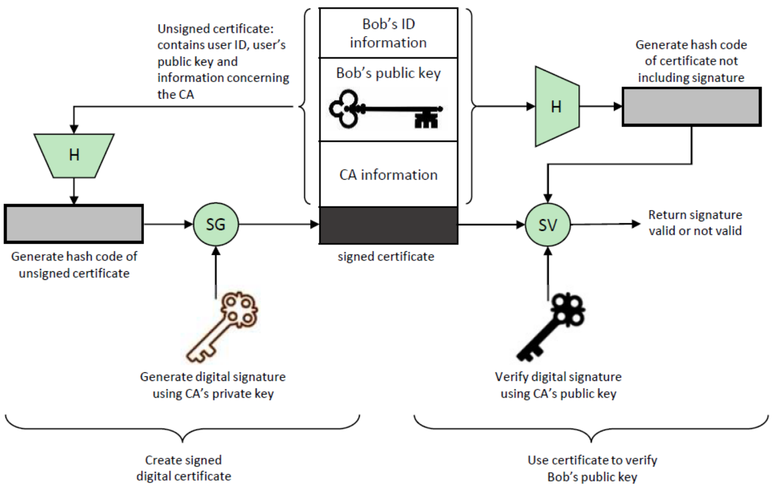
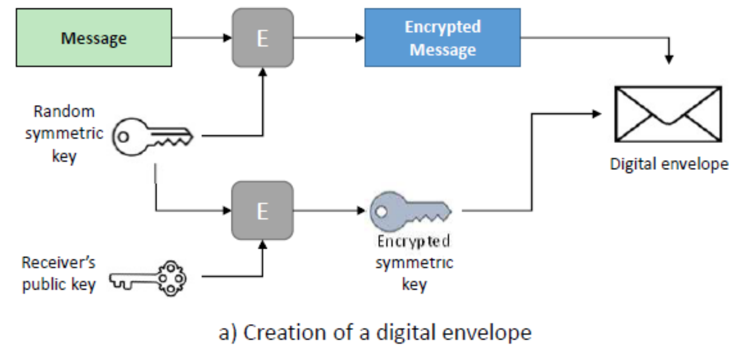

Unsigned certificate: contains user ID, user's public key and information concerning the CA
Bob's ID information
Bob's public key
CA information
H
Generate hash code of certificate not including signature
H
Generate hash code of unsigned certificate
SG
signed certificate
SV
Return signature valid or not valid
Generate digital signature using CA's private key
Verify digital signature using CA's public key
Create signed digital certificate
Use certificate to verify Bob's public key

Figure 9: CA

2

2.3.3.1 Digital envelopes A way to encrypt a message without needing to first arrange for sender and receiver to have the same secret key

[2]CA: certificate authority that is trusted by the user community
government agency, financial institution, . . .



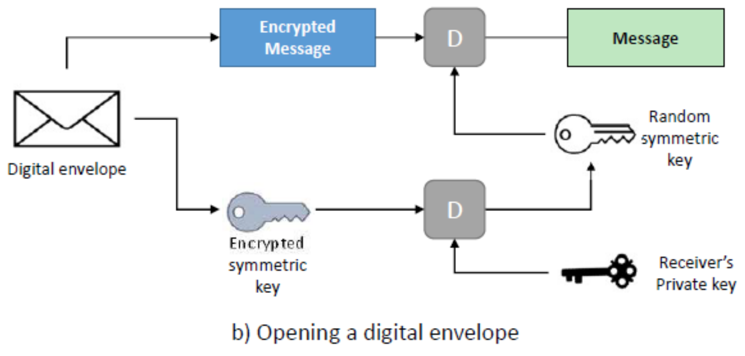


Figure 10

2.4 Random numbers

2.4.1 Uses of random numbers uses

- Keys for public-key algorithms
- Stream key for symmetric stream cipher
- Symmetric key for use as a temporary session key or in creating a digital envelope
- Handshaking to prevent replay attacks
- Session key

2.4.2 Random number requirements

- Randomness
  - Criteria:
    - Uniform distribution
      - Frequency of occurrence of each of the numbers should be approximately the same
    - Independence
      - No one value in the sequence can be inferred from the others
- Unpredictability
  - Each number is statistically independent of other numbers in the sequence
  - Opponent should not be able to predict future elements of the sequence on the basis of earlier elements

2.4.3 Random Vs Pseudorandom

- Cryptographic applications typically make use of algorithms for random number generation
  - Algorithms are deterministic
  - Produce sequences of numbers that are not statistically random

- Pseudorandom numbers are:
  - Sequences produced that satisfy statistical randomness tests
  - Likely to be predictable
- True random number generator (TRNG):
  - Uses a nondeterministic source to produce randomness
  - Most operate by measuring unpredictable natural processes (radiation, gas discharge, leaky capacitors)
  - Increasingly provided on modern processors

2.5 Question & Examples

Questions The schema in figure 8 :

1. provides confidentiality?
2. is Alice sure the message is from Bob?
3. is Alice sure the message has not been altered?

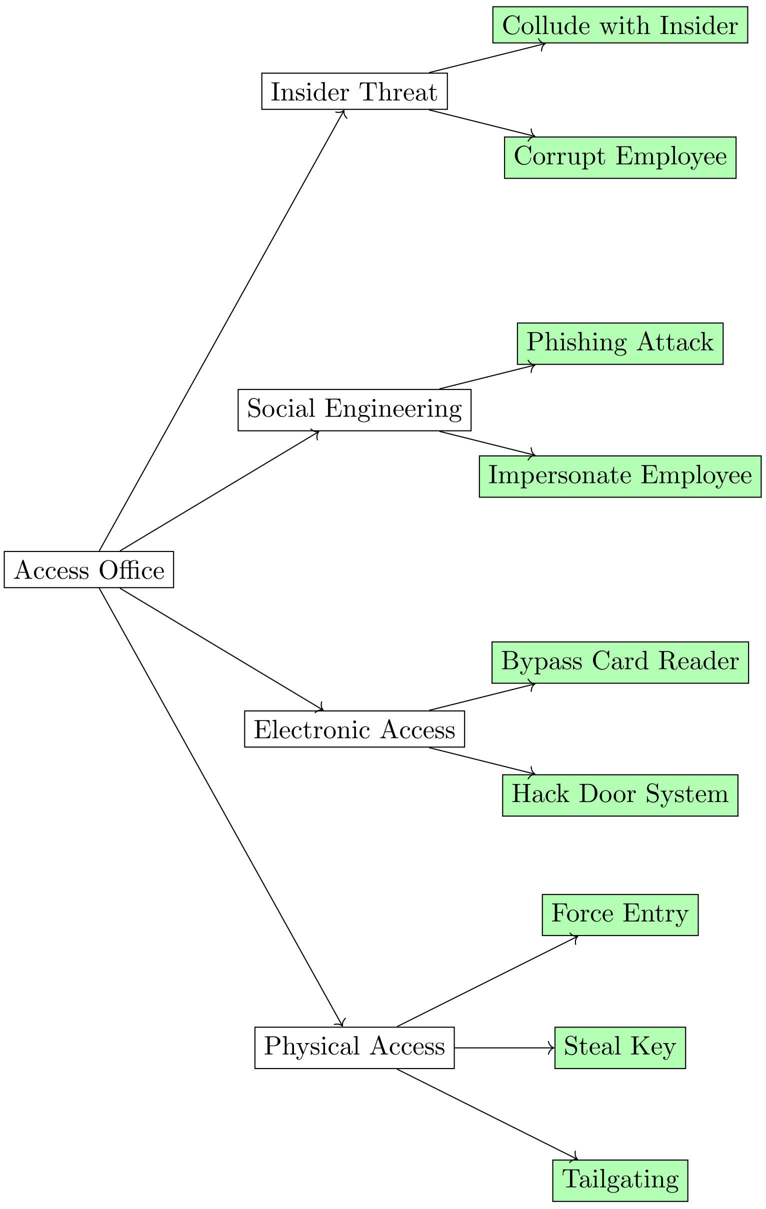
Answers Da vedere

Problem THE SCHEMA (DIGITAL ENVELOPE) IN FIGURE 10 DOES NOT GUARANTEE THE AUTHENTICATION OF THE SENDER. CAN YOU IMPROVE THE SCHEMA TO INCLUDE AUTHENTICATION?

2.6 Exercise

2.6.1 Exercise 1

Question Draw an attack tree for gaining access to a (physical) office.



Answer

2.6.2 Exercise 2

Question $H()$ is a cryptographic hash function that maps a message of an arbitrary bit length onto a 20-bit hash value.

(a) How many random messages would be required to find two different messages M and M' such that $H(M) = H(M')$?

(b) What is the probability that none of n randomly generated messages collide?

Answer Given that H is a cryptographic hash function that maps messages to a 20-bit hash value, we have the following points to consider:

1. **Size of the hash output**: A 20-bit hash value means there are 2^{20} possible hash values. 2. **Birthday problem context**: To find the number of random messages required to find two different messages M and M' such that $H(M) = H(M')$, we can use the birthday paradox.

Part (a): Number of random messages required to find a collision

The birthday paradox states that for a hash function producing n possible values, the approximate number of messages k required to find a collision (i.e., two messages with the same hash value) can be estimated by:

$$k \approx \sqrt{2n}$$

For our 20-bit hash function, $n = 2^{20}$. Plugging this value in:

$$k \approx \sqrt{2 \times 2^{20}} = \sqrt{2^{21}} = 2^{10.5} \approx 2^{10} \times \sqrt{2} \approx 1024 \times 1.414 \approx 1448$$

So, approximately **1448** random messages would be required to find a collision.

Part (b): Probability that none of n randomly generated messages collide

To find the probability that none of n randomly generated messages collide, we need to calculate the probability that all n hash values are unique.

The total number of possible hash values is 2^{20}.

For the first message, there are 2^{20} possible hash values, and the probability of it being unique is 1 (since there are no other messages yet).

For the second message, there are $2^{20} - 1$ possible unique hash values left. Therefore, the probability that the second message does not collide with the first one is:

$$\frac{2^{20} - 1}{2^{20}}$$

For the third message, there are $2^{20} - 2$ possible unique hash values left. The probability that the third message does not collide with the first two is:

$$\frac{2^{20} - 2}{2^{20}}$$

Continuing this pattern, the probability that none of the n messages collide is:

$$P(\text{no collisions}) = \frac{2^{20}}{2^{20}} \times \frac{2^{20} - 1}{2^{20}} \times \frac{2^{20} - 2}{2^{20}} \times \ldots \times \frac{2^{20} - (n-1)}{2^{20}}$$

This can be simplified as:

$$P(\text{no collisions}) = \prod_{i=0}^{n-1} \left(1 - \frac{i}{2^{20}}\right)$$

For large n, this product can be approximated using the exponential function e:

$$P(\text{no collisions}) \approx \exp\left(-\frac{n(n-1)}{2 \times 2^{20}}\right)$$

This approximation is derived from the fact that:

$$\prod_{i=0}^{n-1} \left(1 - \frac{i}{2^{20}}\right) \approx \exp\left(-\sum_{i=0}^{n-1} \frac{i}{2^{20}}\right)$$

And the sum of the first $n - 1$ integers is approximately $\frac{n(n-1)}{2}$.

So, the probability that none of the n messages collide is:

$$P(\text{no collisions}) \approx \exp\left(-\frac{n^2}{2 \times 2^{20}}\right)$$

2.6.3 Exercise 3

Question Alice and Bob are organizing a dinner. Alice is cooking at home and Bob is buying food. To enforce the security in her communications, Alice is appending to her messages a MAC. However, Alice doesn't expect that her messages can be overheard and tampered with by Eve. Alice sends Bob the message $M =$ "buy a gallon of water", with the message authentication code MAC(K,M).

1. Eve intercepts the message and sends to Bob a copy of the same message. How much water will Bob buy? Explain why.
2. Eve intercepts the message and changes "water" with "wine"... will Alice and Bob get drunk tonight?
3. Bob likes beer very much, so he buys beer instead of water and he pretends that the message was $M' =$ "buy a gallon of beer". Is Alice able to prove that the message has been forged by Bob?

Answer

1. Bob will buy a gallon of water, because Eve has sent same message so it hasn't been altered.
2. No, they won't get drunk because if Eve has changed the message also MAC is changed so Bob can check it.
3. No, Alice can prove only that the message has been altered, but not who has modified it.

2.6.4 Exercise 4

Question Same as before, but this time Alice is using a digital signature:
Alice and Bob are organizing a dinner. Alice is cooking at home and Bob is buying food. To enforce the security in her communications, Alice is appending to her messages her digital signature computed with her private key K_A. However, Alice doesn't expect that her messages can be overheard and tampered with by Eve. Alice sends Bob the message $M =$ "buy a gallon of water", with the digital signature $DS(M, K_A)$.

1. Eve intercepts the message and sends to Bob a copy of the same message. How much water will Bob buy? Explain why.
2. Eve intercepts the message and changes "water" with "wine"... will Alice and Bob get drunk tonight?
3. Bob likes beer very much, so he buys beer instead of water and he pretends that the message was $M' =$ "buy a gallon of beer". Is Alice able to prove that the message has/has not been forged by Bob?

Answer

1. Bob will buy a gallon of water, because Eve has sent same message so it hasn't been altered.
2. No, they won't get drunk because Bob is able to check that who has sent the message isn't Alice and this checking the digital signature which isn't the same of orginal message.
3. Yes, Alice can prove that the new message has been forged by Bob, because if the signature has been created by Bob, he would have used his own private key, and it will not match Alice's digital signature.

3 User Identification

3.1 Introduction

3.1.1 Two functions for user authentication

1. User identification
    - by means of a credential or an ID provided by the user to the system
2. User verification
    - by the exchange of authentication information
    - establishes the validity of the claim

3.1.1.1 Digital Authentication Guideline NIST SP 800-63-3 defines digital user authentication as (October 2016) : "*The process of establishing confidence in user identities that are presented electronically to an information system.*"

3.1.1.2 Identification and authentication security requirements

Basic Security Requirements (NIST SP 800-171):

1. Identify users, processes acting on behalf of users, or devices.
2. Authenticate (or verify) the identities of those users, processes, or devices
    - prerequisite to allowing access to organizational information systems.

Derived Security Requirements:

1. Use multifactor authentication for local and network access to privileged accounts and for network access to non-privileged accounts.
2. Employ replay-resistant authentication mechanisms for network access to all accounts.
3. Prevent reuse of identifiers for a defined period.
4. Disable identifiers after a defined period of inactivity.
5. Enforce a minimum password complexity and change of characters when new passwords are created.
6. Prohibit password reuse for a specified number of generations.
7. Allow temporary password use for system logons with an immediate change to a permanent password.
8. Store and transmit only cryptographically-protected passwords.
9. Obscure feedback of authentication information.

3.1.1.3 Four means of authenticating user identity

- Something the individual knows
    - Password, PIN, answers to prearranged questions
- Something the individual possesses (token)
    - Smartcard, electronic keycard, physical key
- Something the individual is (static biometrics)
    - Fingerprint, retina, face
- Something the individual does (dynamic biometrics)
    - Voice pattern, handwriting, typing rhythm

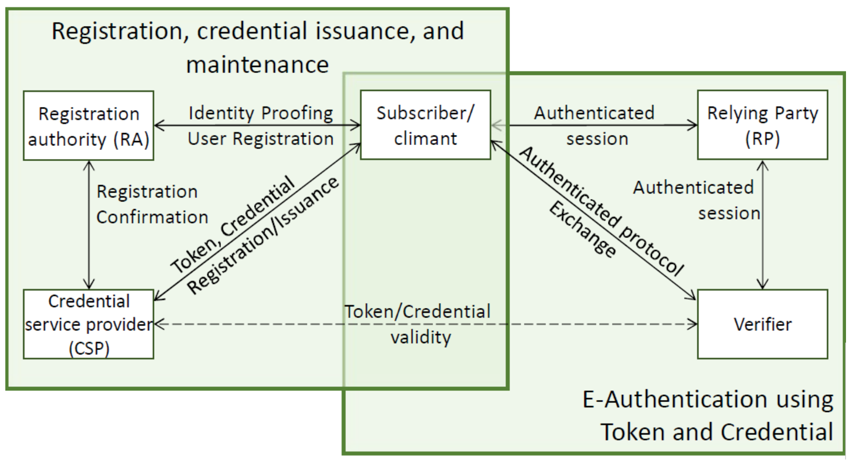

Registration, credential issuance, and maintenance
Registration authority (RA)
Identity Proofing
User Registration
Subscriber/ climant
Authenticated session
Relying Party (RP)
Registration Confirmation
Token, Credential Registration/Issuance
Authenticated protocol Exchange
Authenticated session
Credential service provider (CSP)
Token/Credential validity
Verifier
E-Authentication using Token and Credential

Figure 11: The NIST SP 800-63-3 E-authentication architectural model

3.1.1.4 Risk assessment for user authentication

Assurance level Describes an organization's degree of certainty that a user has presented a credential that refers to his or her identity.
This degree is defined as:

- The degree of confidence in the vetting process used to establish the identity of the individual to whom the credential was issued
- The degree of confidence that the individual who uses the credential is the individual to whom the credential was issued

Potential impact FIPS 199 defines three levels of potential impact on organizations or individuals should there be a breach of security:

- Low
  - An authentication error could be expected to have a limited adverse effect on organizational operations, organizational assets, or individuals
- Moderate
  - An authentication error could be expected to have a serious adverse effect
- High
  - An authentication error could be expected to have a severe or catastrophic adverse effect

Areas of risk Maximum potential impact for each assurance level

	Assurance Level Impact Profiles			
Potential Impact Categories for Authentication Errors	**1**	**2**	**3**	**4**
Inconvenience, distress, or damage to standing or reputation	Low	Mod	Mod	High
Financial loss or organization liability	Low	Mod	Mod	High
Harm to organization programs or interests	None	Low	Mod	High
Unauthorized release of sensitive information	None	Low	Mod	High
Personal safety	None	None	Low	Mod/High
Civil or criminal violations	None	Low	Mod	High

Table 3: Assurance Level Impact Profiles

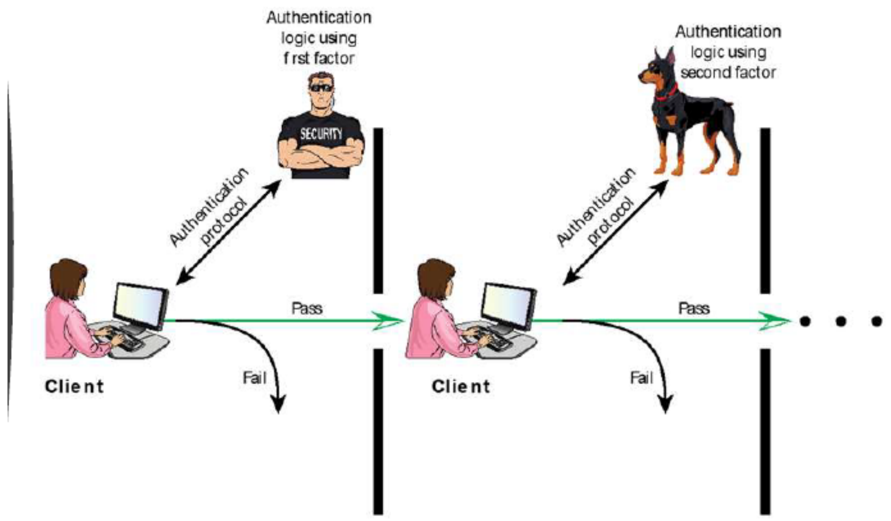


Figure 12: Multifactor authentication

3.1.2 Question & Example

Question: REFERRING TO THE NIST SP 800-63-3 MODEL, DID YOU EVER EXPERIENCE A CASE OF AUTHENTICATION THAT FOLLOWS THAT MODEL? DESCRIBE THAT AUTHENTICATION SYSTEM FROM THE USER PERSPECTIVE.

Answers: Yes, I have experienced an authentication system that follows the NIST SP 800-63-3 model. The system is designed to ensure secure and reliable authentication processes. Below is a description of the authentication system from the user's perspective:

- **Initial Registration:**
  - The user begins by visiting the service's registration page.
  - The user is required to provide personal information, such as full name, email address, and phone number.
  - The user sets a password that meets the specified complexity requirements.
  - The system sends a verification email to the provided email address. The user must click on the link in the email to verify their email address.
  - Optionally, the user may be asked to provide additional identity proofing documents or answer knowledge-based questions for higher assurance levels.
- **Authentication:**
  - The user navigates to the login page of the service.
  - The user enters their username (or email) and password.
  - For multi-factor authentication (MFA), the user is prompted to enter a code sent to their registered phone number via SMS, or to approve the login attempt using an authentication app.
  - Upon successful verification of the provided factors, the user is granted access to their account.
- **Session Management:**
  - Once authenticated, the user can manage their session and perform actions within their account.
  - The system may prompt the user to re-authenticate for sensitive actions, such as changing account settings or accessing personal data.
- **Account Recovery:**

  - If the user forgets their password, they can initiate a password recovery process.
  - The system sends a password reset link to the user's registered email address.
  - The user clicks the link and follows the instructions to set a new password.
  - Additional verification steps, such as answering security questions or using MFA, may be required to complete the password reset process.

- **Periodic Verification:**
  - Periodically, the system may prompt the user to re-verify their email address and phone number to ensure the information is current.
  - The user may also be required to review and update their security settings.

This multi-layered approach to authentication, incorporating both knowledge-based and possession-based factors, aligns with the NIST SP 800-63-3 guidelines to provide a high level of assurance in the authentication process.

3.2 Password-Based Authentication

- Widely used line of defense against intruders
  - User provides name/login and password
  - System compares password with the one stored for that specified login
- The user ID:
  - Determines that the user is authorized to access the system
  - Determines the user's privileges
  - Is used in discretionary access control

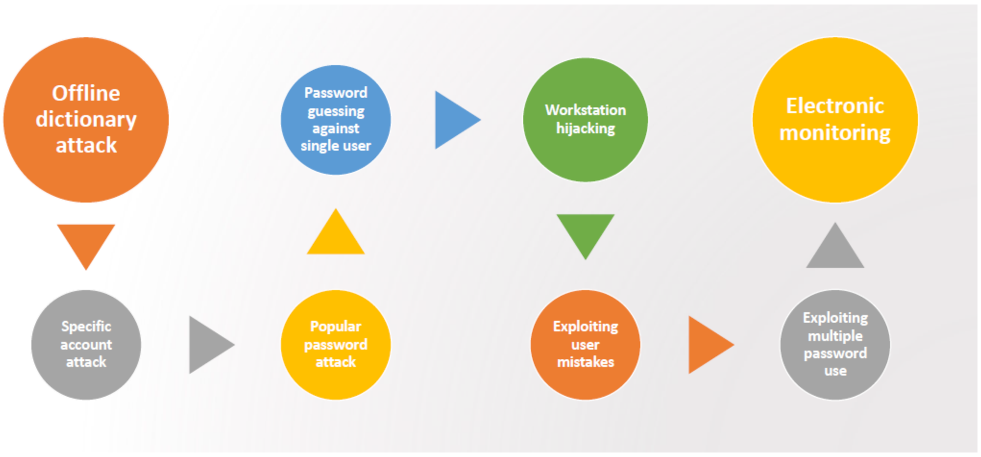


Figure 13: Password Vulnerabilities

3.2.1 Unix Implementation

The salt serves three purposes:

- It prevents duplicate passwords from being visible in the password file.
- It becomes nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them.
- It greatly increases the difficulty of offline dictionary attacks.
  - Say the attacker obtains a copy of the password file and salt is not used
    * . . . and its goal is to guess a single password of any user.
  - The attacker may submit a large number of likely passwords to the hashing function:
    * If any of the guesses matches one of the hashes in the file, then the attacker has found a password that is in the file.
  - Using salt instead, the attacker must take each guess and submit it to the hash function once for each salt value in the dictionary file, multiplying the number of guesses that must be checked.

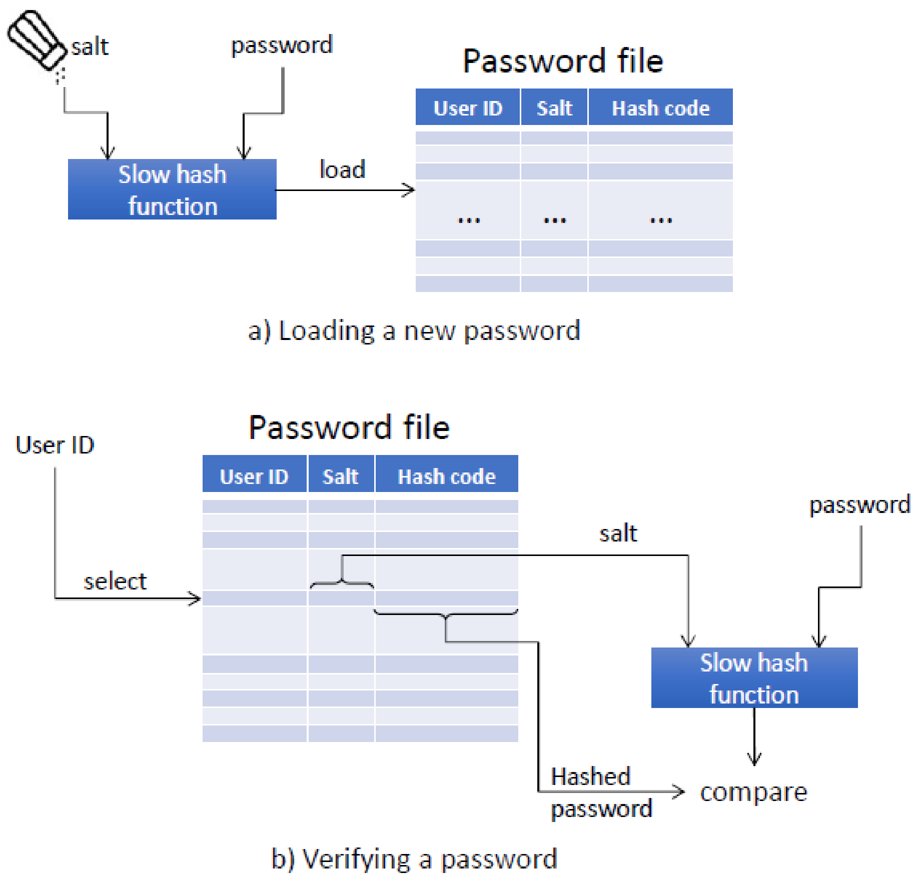


Figure 14: Unix Scheme

3.2.1.1 Implementation

Original Scheme

- Up to eight printable characters in length
- 12-bit salt used to modify DES encryption into a one-way hash function
- Repeatedly encrypted 25 times
- Output translated to 11 character sequence

Now regarded as inadequate still often required for compatibility with existing account management software or multivendor environments

Improved implementations

1. Much stronger hash/salt schemes available for Unix
2. Recommended hash function is based on MD5
   - Salt of up to 48-bits
   - Password length is unlimited
   - Produces 128-bit hash
   - Uses an inner loop with 1000 iterations to achieve slowdown
3. OpenBSD uses Blowfish block cipher-based hash algorithm called Bcrypt
   - Most secure version of Unix hash/salt scheme
   - Uses 128-bit salt to create 192-bit hash value

3.2.2 Password Cracking

- Dictionary attacks
  - Develop a large dictionary of possible passwords and try each against the password file
  - Each password must be hashed using each salt value and then compared to stored hash values
- Rainbow table attacks
  - Pre-compute tables of hash values of passwords in dictionary for all salts
  - A mammoth table of hash values
  - Can be countered by using a sufficiently large salt value and a sufficiently large hash length
- Password crackers exploit the fact that people choose easily guessable passwords
  - Shorter password lengths are also easier to crack
- John the Ripper
  - Open-source password cracker first developed in 1996
  - Uses a combination of brute-force and dictionary techniques

3.2.2.1 Modern Approaches

- Complex password policy
  - Forcing users to pick stronger passwords
- However, password-cracking techniques have also improved
  - The processing capacity available for password cracking has increased dramatically
  - The use of sophisticated algorithms to generate potential passwords
  - Studying examples and structures of actual passwords in use

3.2.2.2 Password file access control Can block offline guessing attacks by denying access to encrypted passwords

- Make available only to privileged users
- Shadow Password File

Vulnerabilities

- Weakness in the OS that allows access to the file
- Accident with permissions making it readable
- Users with same password on other systems
- Access from backup media
- Sniff passwords in network traffic

3.2.2.3 Password selection strategies

1. User education:
   - Users can be told the importance of using hard to guess passwords and can be provided with guidelines for selecting strong passwords.
2. Computer generated passwords:
   - Users have trouble remembering them.
3. Reactive password checking:
   - System periodically runs its own password cracker to find guessable passwords.
4. Complex password policy:
   - User is allowed to select his own password, but the system checks if the password is allowable. If not, rejects it.
   - Goal is to eliminate guessable passwords while allowing the user to select a password easy to remember.

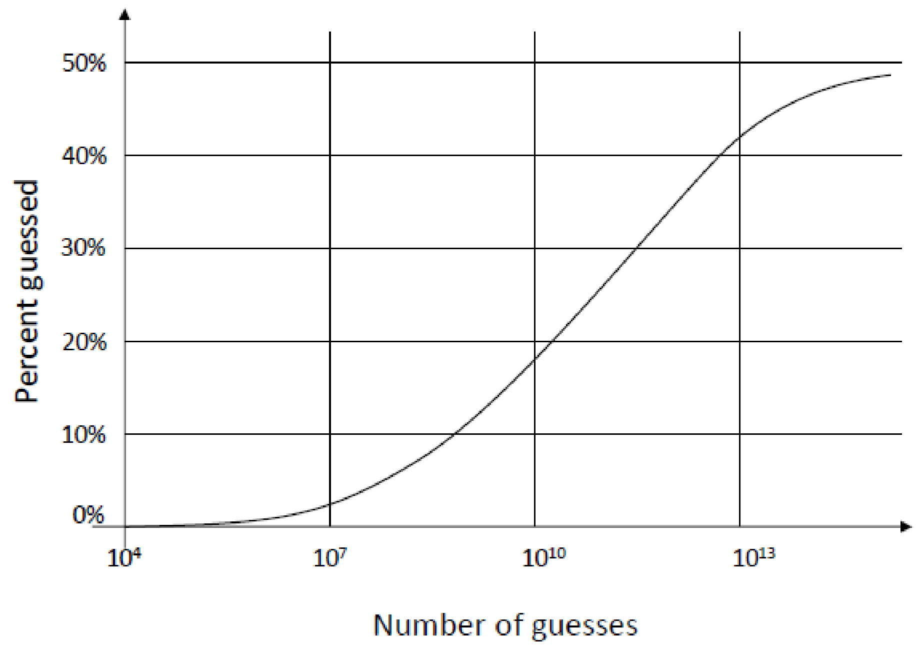


Figure 15: Percentage of passwords guess after a given number of guesses

3.2.2.4 Proactive password checking

1. Rule enforcement:
   - Specific rules that passwords must adhere to
2. Password checker
   - Compile a large dictionary of "bad" passwords not to use
3. But how to make a fast password check?
   - Bloom filter, used to implement password checkers
     - Exploits k hash functions to build tables of hash values
     - Check desired password against this table
     - It's a probabilistic, efficient way to check if a password is in the dictionary of forbidden passwords

Bloom filter password checker: Bloom filter constructed over a dictionary $\mathcal{D}$ of passwords... say that:

- $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$
- $h_1, h_2, \ldots, h_k$ are hash functions, where $h_i : \mathcal{D} \rightarrow [0, n-1]$
- Bloom filter $\mathcal{B}$ is an array of n bits

Bloom filter constructed as:

```
Let $\mathcal{B}[i] = 0$ for each $i \in 0, 1, \ldots, n-1$ :
For each $x \in \mathcal{D}$ :
    For each $j \in 1, 2, \ldots, k$ :    $\mathcal{B}[h_j(x)] = 1$
```

To check a password y with the Bloom filter:

```
if $\mathcal{B}[h_j(y)] = 0$ for some $j \in 1, 2, \ldots, k$ :
    return valid    //password $y$ is not in $\mathcal{D}$
else:
    return rejected    //password $y$ may be in $\mathcal{D}$
```

The Bloom filter does not have false negatives:
*If **valid**, then y is not present in $\mathcal{D}$*
The Bloom filter may have false positives:
*If **rejected**, then y may still not be in $\mathcal{D}$*

Example: $\mathcal{D} = \{\text{Stefano}, \text{Paolo}\}$
Say that:

$$h_1(\text{Stefano}) = 10; \quad h_2(\text{Stefano}) = 7; \quad h_3(\text{Stefano}) = 3$$

$$h_1(\text{Paolo}) = 7; \quad h_2(\text{Paolo}) = 12; \quad h_3(\text{Paolo}) = 1$$

Hence $\mathcal{B} =$

0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	0	1	0	0	0	1	0	0	1	0	1

Now, consider passwords Rita and Sofia, and that:

$$h_1(\text{Rita}) = 3; \quad h_2(\text{Rita}) = 12; \quad h_3(\text{Rita}) = 1$$

$$h_1(\text{Sofia}) = 2; \quad h_2(\text{Sofia}) = 9; \quad h_3(\text{Sofia}) = 7$$

Then:

- Rita is rejected
- Sofia is valid

Need to find a proper configuration of the parameters (values of k, m, n), else the mechanism may be unusable it may give too many false positives

3.2.3 Question & Examples

Question: AN ADVERSARY PERFORMS A BRUTE FORCE ATTACK AND IN 1 YEAR TESTS EXHAUSTIVELY ALL THE POTENTIAL PINS OF A USER. IF THE PINS ARE OF 6 DIGITS, EACH IN AN ALPHABET OF 10 SYMBOLS, HOW LONG IT TAKES TO TEST ONE PIN?

Answer:

- Each PIN is 6 digits long.
- The alphabet for each digit consists of 10 symbols (0-9).
- Therefore, the total number of possible PINs is:

$$10^6 = 1,000,000$$

- The adversary tests all 1,000,000 possible PINs in 1 year.
- The duration of 1 year can be converted to seconds as follows:

$$\begin{aligned} 1 \text{ year} &= 365 \,\text{days} \\ &= 365 \times 24 \,\text{hours} \\ &= 365 \times 24 \times 60 \,\text{minutes} \\ &= 365 \times 24 \times 60 \times 60 \,\text{seconds} \\ &= 31,536,000 \,\text{seconds} \end{aligned}$$

- The time taken to test one PIN can be calculated by dividing the total time by the number of PINs:

$$\text{Time per PIN} = \frac{\text{Total time in seconds}}{\text{Total number of PINs}}$$

$$\text{Time per PIN} = \frac{31,536,000\,\text{seconds}}{1,000,000} \approx 31.536\,\text{seconds per PIN}$$

Thus, it takes approximately 31.536 seconds to test one PIN.

3.2.4 Question & Examples

Question Comments on the Suitability of Passwords

Answer :

- **Back#To#Black#**
  This password contains a mix of uppercase and lowercase letters, special characters (#), and has a relatively good length. It is generally considered a strong password due to its complexity and length. However, the repeated pattern of special characters might make it slightly predictable. Overall, it is still a good choice.
- **09876**
  This password is entirely numeric and only 5 characters long. It is very weak as it can be easily guessed or cracked using brute force attacks. Short numeric passwords are generally not recommended due to their low entropy.
- **viaFermi3**
  This password contains lowercase letters and a digit, making it moderately strong. It has a reasonable length and includes a mix of characters. However, adding uppercase letters and special characters could further improve its strength.
- **ketchup**
  This password is a common word and entirely lowercase, making it weak. It lacks complexity and is susceptible to dictionary attacks. Passwords based on common words are not recommended.
- **onafetS**
  This password contains only letters but has a mix of uppercase and lowercase characters. It is relatively short and could be stronger with the inclusion of digits and special characters. Additionally, it might be a reversal of a common word, which could make it more predictable.
- **S0Ll3van7e**
  This password is strong as it includes uppercase and lowercase letters, digits, and has a good length. The mix of different character types and its length contribute to higher entropy, making it a good choice for a secure password.

Question THE SALT IN THE UNIX PASSWORD SCHEME INCREASES THE DIFFICULTY OF GUESSING BY A FACTOR OF 4096.

1. HOW MANY BITS IS THE SALT?
2. THE SALT IS STORED IN PLAINTEXT IN THE SAME ENTRY AS THE CORRESPONDING CIPHERTEXT PASSWORD. THEREFORE, IT IS KNOWN TO THE ATTACKER AND NEED NOT BE GUESSED. WHY IS IT ASSERTED THAT THE SALT INCREASES SECURITY?

Answer

1. The salt is made by 12 bit, that is saved with User ID and Ciphertext Password, that permit to hash result to be more random.
2. The salt increases security because ,by changing it , allowed to have different hash result also if password used is the same.

Question STILL ABOUT SALT: WOULDN'T IT BE POSSIBLE TO THWART COMPLETELY ALL PASSWORD CRACKERS BY DRAMATICALLY INCREASING THE SALT SIZE TO, SAY, 24 OR 48 BITS?

Answer Increasing the salt size, surelly, we raise password's security, but to thwart completely all password cracker is impossible because there are many techniques that permit to semply also brute force as Dictionary.

3.3 Token-Based authentication

Objects that a user possesses for the purpose of user authentication are called tokens:

- Memory cards
- Smart tokens/cards

Card Type	Defining Feature	Example
Embossed	Raised characters only, on front	Old credit card
Memory	Electronic memory inside	Prepaid phone card
Smart tokens	Electronic memory and processor inside	
Contact	Electrical contacts exposed on surface	Biometric ID card
Contactless	Radio antenna embedded inside	

Table 4: Types of cards used as tokens

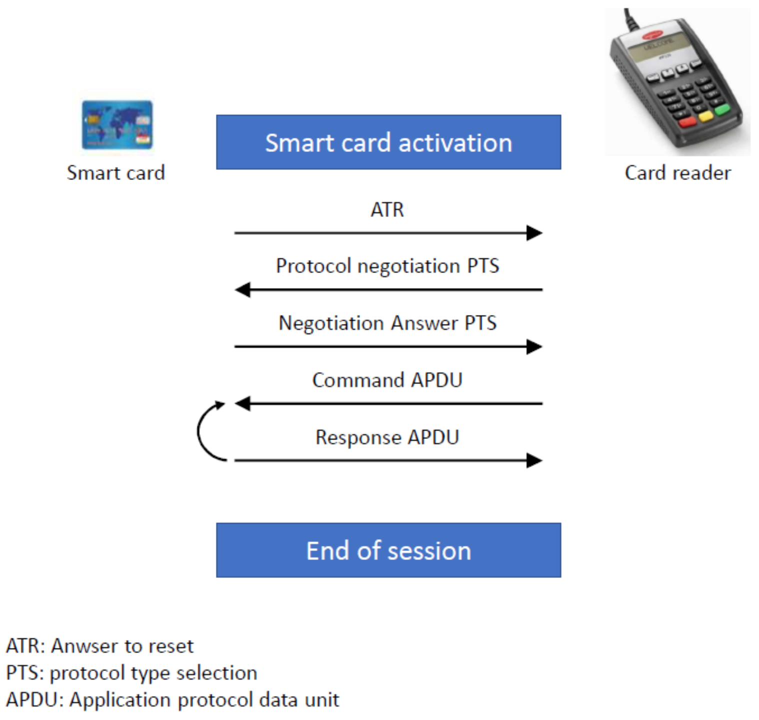


Figure 16: Smart cart / reader exchange

3.3.1 Application

3.3.1.1 Electronic Identity Cards (eID) Use of a smart card as a national identity card for citizens. Can serve the same purposes as other national ID cards, and similar cards such as a driver's license, for access to government and commercial services. Can provide stronger proof of identity and can be used in a wider variety of applications. In effect, is a smart card that has been verified by the national government as valid and authentic. An example is the German card neuer Personalausweis. Has human-readable data printed on its surface

- Personal data
- Document number

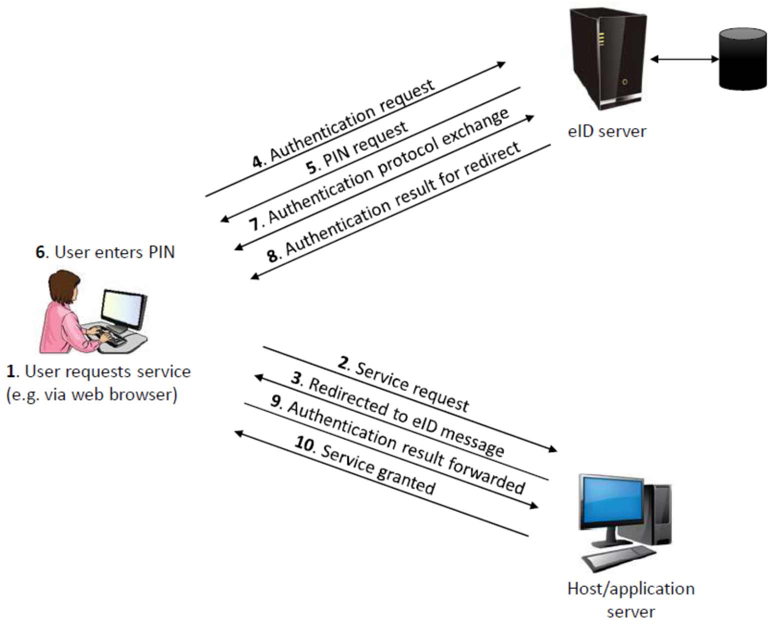

4. Authentication request
5. PIN request
7. Authentication protocol exchange
8. Authentication result for redirect
eID server
6. User enters PIN
1. User requests service (e.g. via web browser)
2. Service request
3. Redirected to eID message
9. Authentication result forwarded
10. Service granted
Host/application server

Figure 17: User authentication with eID

- Card access number (CAN)
- Machine readable zone (MRZ)

Function	**Purpose**	**PACE Password**	**Data**	**Users**
ePass (mandatory)	Authorized offline inspection systems read the data	CAN or MRZ	Face image; two fingerprint images (optional); MRZ data	Offline biometric identity verification reserved for government access
eID (activation optional)	Online applications read the data or access functions as authorized.	eID PIN	Family and given names; artistic name and doctoral degree: date and place of birth; address and community ID; expiration date	Identification; age verification; community ID verification; restricted identification (pseudonym); revocation query Offline inspection systems read the
	Offline inspection systems read the data and update the address and community ID.	CAN or MRZ		
eSign (certificate optional)	A certification authority installs the signature certificate online.	eID PIN	Signature key; X.509 certificate	Electronic signature Citizens make signature creation
	Citizens make signature creation creation electronic signature with eSign PIN.	CAN		

Table 5: Electronic functions for eID cards

CAN card access number

MRZ machine readable zone

PACE password authenticated connection establishment

PIN personal identification number

3.3.1.2 Password Authenticated Connection Establishment

The eID card has a contactless RF chip that is protected by access control measures to ensure it cannot be read without proper authorization.

- **Online applications**: The user must enter a 6-digit PIN, which should only be known to the cardholder.

- **Offline applications**: Access is granted using either the MRZ (machine-readable zone) printed on the back of the card or the CAN (card access number) printed on the front.

Explanation:

- **Contactless RF Chip Protection**: The eID card contains a contactless radio frequency (RF) chip. To protect the data on this chip, explicit access control mechanisms are in place. This means the chip cannot be read unless the correct access protocols are followed.
- **Online Applications**: When using the eID card for online purposes, the user needs to enter a 6-digit personal identification number (PIN). This PIN acts as a security measure, ensuring that only the person who knows the PIN (presumably the cardholder) can use the card.
- **Offline Applications**: In scenarios where the eID card is used offline (e.g., at a physical terminal), access to the card's data can be gained by using either:
  - The **MRZ** (machine-readable zone), which is a string of characters printed on the back of the card that can be read by a machine.
  - The **CAN** (card access number), which is a specific number printed on the front of the card.

3.4 Biometric Authentication

Attempts to authenticate an individual based on unique physical characteristics, it is based on pattern recognition and is technically complex and expensive when compared to passwords and tokens **Physical characteristics usde include:**

- Facial characteristics
- Fingerprints
- Hand geometry
- Retinal pattern
- Iris
- Signature
- Voice

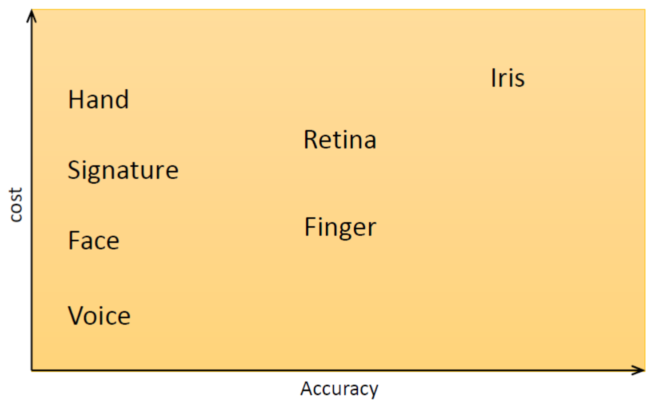


Figure 18: Cost vs accuracy of biometric characteristics in user authentication schemes

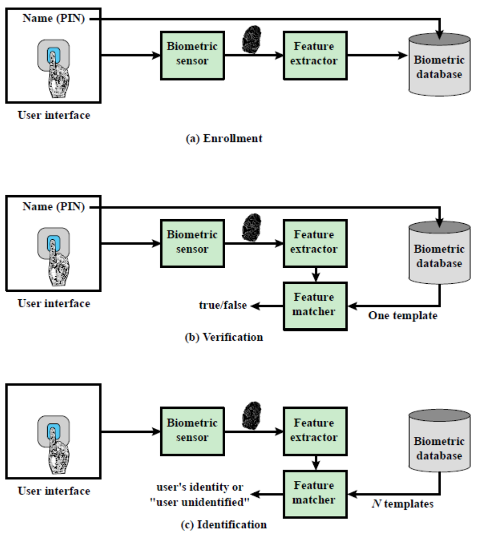


Figure 19: A generic biometric system

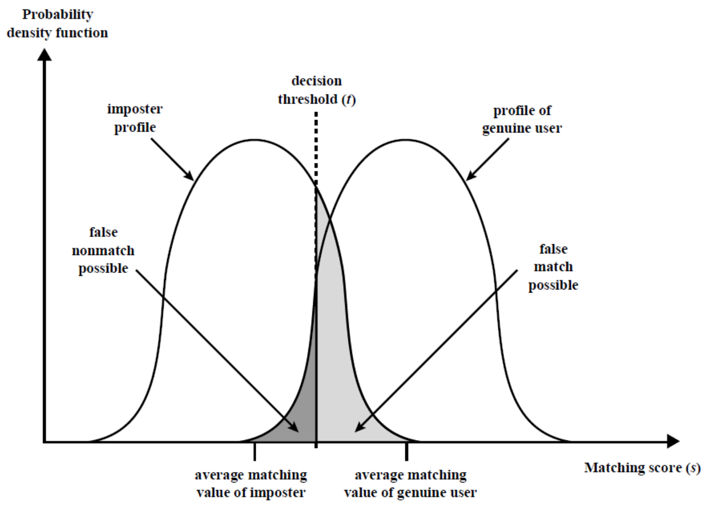


Figure 20: Profiles of biometric characteristics

3.4.1 Question & Examples

Question: "JANE SPLIT A STONE INTO TWO PIECES, KEPT ONE FOR HERSELF, AND GAVE THE OTHER TO JASON. SO SHE SAID - WHEN YOU WILL SEND YOUR EMISSARY GIVE HIM THIS HALF OF THE STONE AND I WILL RECOGNIZE HIM." WHAT KIND OF AUTHENTICATION IS THIS?

Answer This scenario describes a form of **physical token-based authentication**. In this type of authentication, the possession of a specific physical object (in this case, the matching half of a split stone) is used to verify the identity of an individual. The stone acts as a unique token that is difficult to replicate, ensuring that only someone with the corresponding half of the stone can be authenticated as the legitimate emissary.

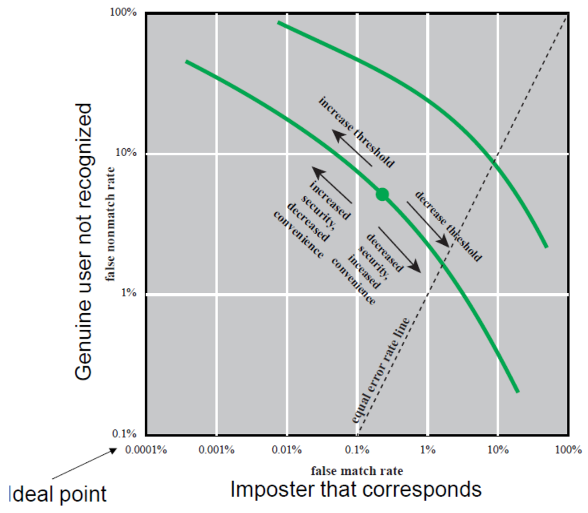


Figure 21: Idealized biometric measurement operating characteristic curves

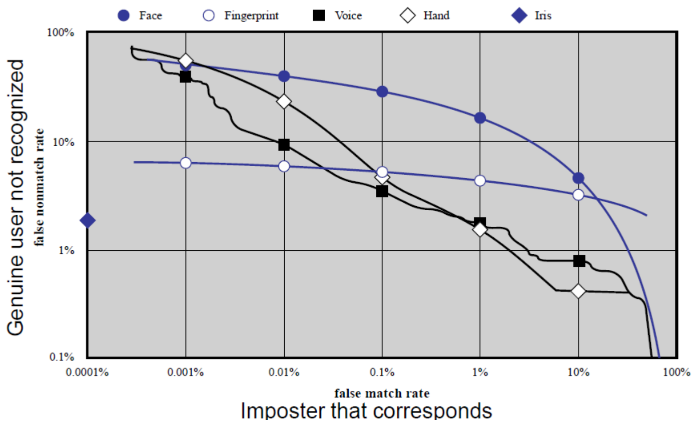
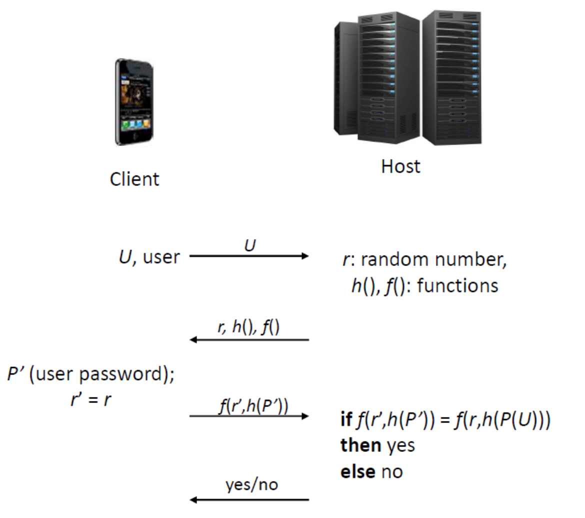


Figure 22: Actual biometric measurement operating characteristic curves (log-scale)

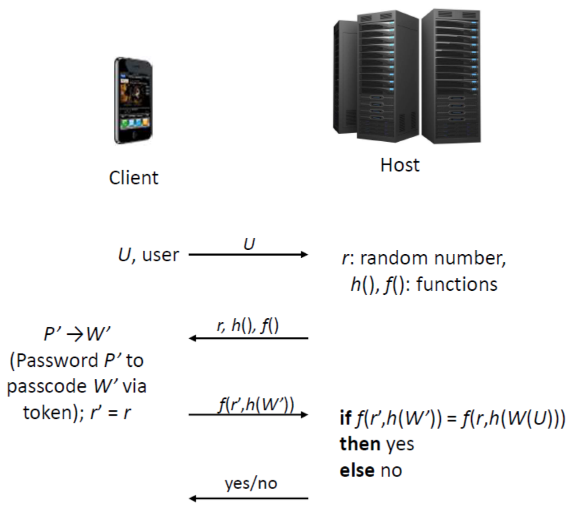
3.5 Remote User Authentication

Authentication over a network, the Internet, or a communications link is more complex because there are additional security threats such as: **Eavesdropping, capturing a password, replaying an authentication sequence that has been observed** and generally rely on some form of a challengeresponse protocol to counter threats.

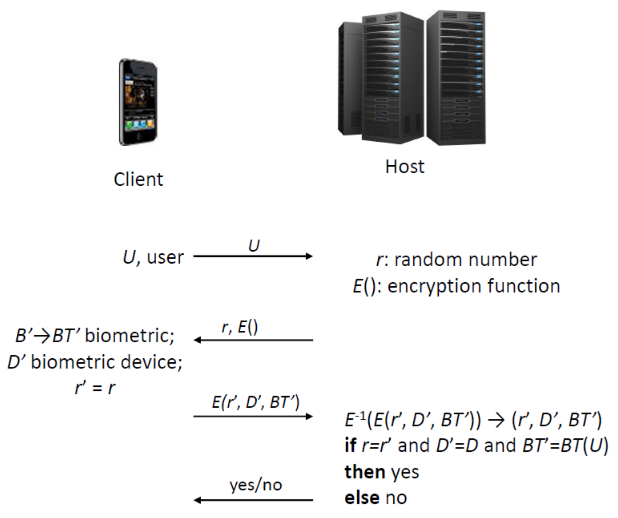
3.5.1 Basic challengeresponse protocol for remote user authentication



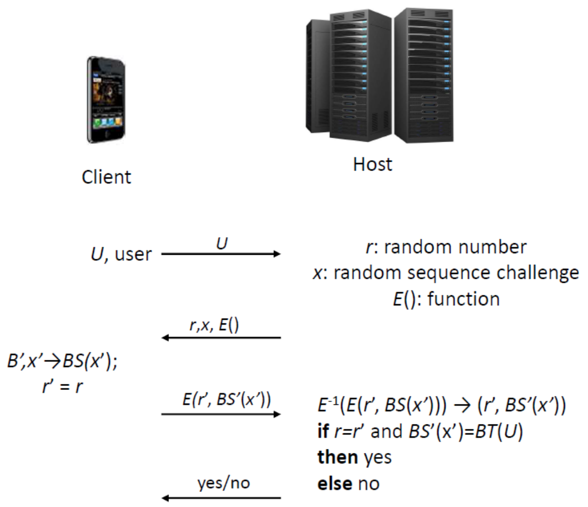
a) Challenge-response protocol for authentication via password



b) Authentication protocol with token



c) Authentication protocol with static biometric



d) Authentication protocol with dynamic biometric

3.5.2 Authentication security issues

Eavesdropping Adversary attempts to learn the password by some sort of attack that involves the physical proximity of user and adversary.

Host Attacks Directed at the user file at the host where passwords, token passcodes, or biometric templates are stored.

Replay Adversary repeats a previously captured user response.

Client Attacks Adversary attempts to achieve user authentication without access to the remote host or the intervening communications path.

Trojan Horse An application or physical device masquerades as an authentic application or device for the purpose of capturing a user password, passcode, or biometric.

Denial-of-Service Attempts to disable a user authentication service by flooding the service with numerous authentication attempts.

Table 6: Some potential attacks, susceptible authenticators, and typical defenses

Attacks	**Authenticators**	**Examples**	**Typical Defenses**
Client attack	Password	Guessing, exhaustive search	Large entropy; limited attempts
	Token	Exhaustive search	Large entropy; limited attempts; theft of object requires presence
	Biometric	False match	Large entropy; limited attempts
Host attack	Password	Plaintext theft, dictionary/exhaustive search	Hashing; large entropy; protection of password database
	Token	Passcode theft	Same as password; 1-time passcode
	Biometric	Template theft	Capture device authentication; challenge response
Eavesdropping, theft, and copying	Password	"Shoulder surfing"	User diligence to keep secret; administrator diligence to quickly revoke compromised passwords; multifactor authentication
	Token	Theft, counterfeiting hardware	Multifactor authentication; tamper resistant/evident token
	Biometric	Copying (spoofing) biometric	Copy detection at capture device and capture device authentication
Replay	Password	Replay stolen password response	Challenge-response protocol
	Token	Replay stolen passcode response	Challenge-response protocol; 1-time passcode
	Biometric	Replay stolen biometric template response	Copy detection at capture device and capture device authentication via challenge-response protocol
Trojan horse	Password, token, biometric	Installation of rogue client or capture device	Authentication of client or capture device within trusted security perimeter
Denial of service	Password, token, biometric	Lockout by multiple failed authentications	Multifactor with token

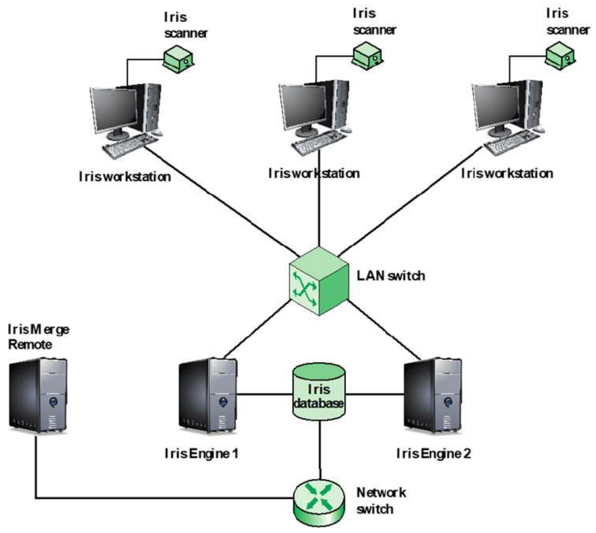


Figure 23: General iris scan site architecture for UAE system

3.6 Exercises

3.6.1 Exercise 1

Question: A system requests the users to choose passwords at least 8 and at most 10 characters long and that are chosen within an alphabet of 40 symbols. The system combines the passwords with a 10 bits salt to produce a hash code for each password that is stores, along with the salt, in the password file. The hash is 128 bits long. Assume also that testing a password takes 0.1 milliseconds.

1. An adversary that got access to the password file performs a brute force attack to crack the password of a specific user. How long it will take in the worst case and on average?
2. Assume the system has 10,000 users, and the adversary is interested in breaking the password of one arbitrary user (any user would be OK to get in). How long it will take on average? How long it would take if no salt was used?

Answer:

1. Response:

   $S =$ Number of Symbols $= 40, N =$ lengths of password $= 8, 9, 10, P =$ Number of Permutation$, C =$ Permutation with sa

   - The number of different passwords are: To calculate this number we will use Permutation with Repetition

     $$P = 40^8 + 40^9 + 40^{10} = 1.075 * 10^{16}$$

   - Each password is combined with a salt randomly chosen in combinations:

     $$CS = 1024$$

   - Thus the number of combinations to be generated is:

     $$C = P * CS = 1,10 * 10^{19}$$

   - In the worst case it will take: To taka worst case we consider all combination, so first we transfororm milliseconds in seconds

     $$SEC = 0.1/1000 = 10^{-4}$$

     then we calculate number of second necessarily

     $$N_{SEC} = C * SEC$$

     and finally transform its in year

     $$N_Y = N_{SEC}/SEC_Y = 3.492.794, 2years$$

- On average it will take: you just need to divide by two -¿

$$NA_{SEC} = N_{SEC}/2$$

and after

$$N_Y = NA_{SEC}/SEC_y$$

2. Response:
   - The number of different passwords are: 1.101.511.929.856.000.000

$$C = \text{like first question}$$

   - Each password is combined with a salt randomly chosen in 1024 combinations.
   - On average: To calculate it,you have to multiply

$$C * 10.000, \text{Then to make same calculations made for first Question} andresultis 17.466.263$$

   years to crack the password of one arbitrary user in a system with 10.000 users.
   - Thus it will take on average: 17.466.263 years
   - If no salt was used it will take on average: 17.048 years because we have only permutation

$$P$$

   as number of passwords.

Question: A system requests the users to choose passwords at least 8 and at most 10 characters long and that are chosen within an alphabet of 40 symbols. The system combines the passwords with a 10 bits salt to produce a hash code for each password that is stores, along with the salt, in the password file. The hash is 32 bits long. Assume also that testing a password takes 0.1 milliseconds. An adversary that got access to the password file performs a brute force attack to crack the password of a specific user. How long it will take in the worst case and on average?

Answer:

- The number of different passwords are: Same value of previous exercise

$$P$$

- Each password is combined with a salt randomly chosen in 1024 combinations because is composed by 10 bits
- The number of different hashes in which the password is encoded is: 4.294.967.296
- On average, the number of combinations to be generated is:
- On average it will take:

4 Access Control

4.1 Introduction

4.1.0.1 Definition

NISTIR 7298: "the process of granting or denying specific requests to:

(1) obtain and use information and related information processing services; and
(2) enter specific physical facilities"

RFC 4949: "a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy"

4.1.0.2 Security Requirements

1. Limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems).
2. Limit information system access to the types of transactions and functions that authorized users are permitted to execute.
3. Control the flow of CUI [3] (controlled unclassified information) in accordance with approved authorizations.
4. Separate the duties of individuals to reduce the risk of malevolent activity without collusion.
5. Employ the principle of least privilege, including for specific security functions and privileged accounts.
6. Use non-privileged accounts or roles when accessing non-security functions.
7. Prevent non-privileged users from executing privileged functions and audit the execution of such functions.
8. Limit unsuccessful logon attempts.
9. Provide privacy and security notices consistent with applicable CUI rules.
10. Use session lock with pattern-hiding displays to prevent access and viewing of data after period of inactivity.
11. Terminate (automatically) a user session after a defined condition.
12. Monitor and control remote access sessions.
13. Employ cryptographic mechanisms to protect the confidentiality of remote access sessions.
14. Route remote access via managed access control points.
15. Authorize remote execution of privileged commands and remote access to security-relevant information.
16. Authorize wireless access prior to allowing such connections.
17. Protect wireless access using authentication and encryption.
18. Control connection of mobile devices.
19. Encrypt CUI on mobile devices.
20. Verify and control/limit connections to and use of external information systems.
21. Limit use of organizational portable storage devices on external information systems.
22. Control CUI posted or processed on publicly accessible information systems.

4.1.0.3 Principles In a broad sense, all of computer security is concerned with access control RFC 4949 defines computer security as:

> "measures that implement and assure security services in a computer system, particularly those that assure access control service"

4.1.0.4 Computer Security We address a specific concept of access control, that implements a security policy, it specifies who (or what) may have access to each specific system resource, and the type of access that is permitted in each instance

4.1.0.5 Other Security Functions In addition to access control, this context involves the following entities and functions:

Authentication: Verification that the credentials of a user or other system entity are valid.

Authorization: The granting of a right or permission to a system entity to access a system resource. This function determines who is trusted for a given purpose.

Audit: An independent review and examination:

- checks the adequacy of system controls,
- ensures compliance with established policy and operational procedures,
- detects breaches in security,
- recommends any indicated changes in control, policy and procedures.

[3] CUI : controlled unclassified information

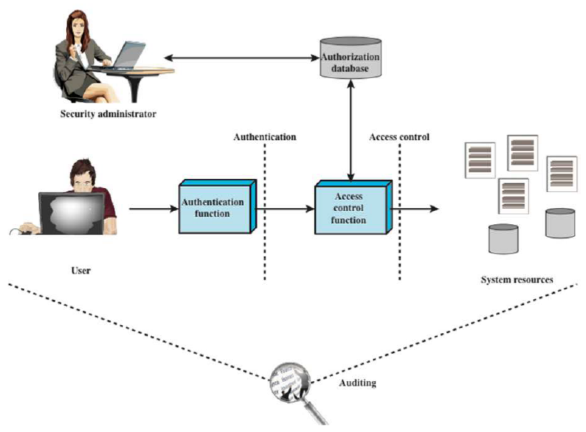


Figure 24: Security Functions

4.1.0.6 Policies

Discretionary access control (DAC):
- Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do.

Mandatory access control (MAC):
- Controls access based on comparing security labels with security clearances.

Role-based access control (RBAC):
- Controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles.

Attribute-based access control (ABAC):
- Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions.

4.1.0.7 Subjects, objects, and access rights

Subject: An entity capable of accessing objects

- Generally, the concept of subject equates with that of process.
- A user/application accesses objects by means of a process that represents it.
- The process takes on the attributes of the user/application, such as access rights.
- It is held accountable for its actions, an audit trail may be used to record the association of a subject with security relevant actions performed on an object by the subject.

classes of subjects: **Owner:** The creator of a resource (a system administrator or a project leader…).
Group: Membership in the group lets to exercise some access rights. A user may belong to multiple groups.
World: Users that are able to access the system anyway. Grants the least access rights necessary.

Object: A resource to which access is controlled

- Contains and/or receives information:
  - records, blocks, pages, files, portions of files, directories, messages, programs etc.
  - may also be: bits, bytes, words, processors, ports, clocks, etc.
- Number and types depends on:
  - the context in which access control operates
  - the tradeoff between security vs complexity, processing burden, and ease of use.

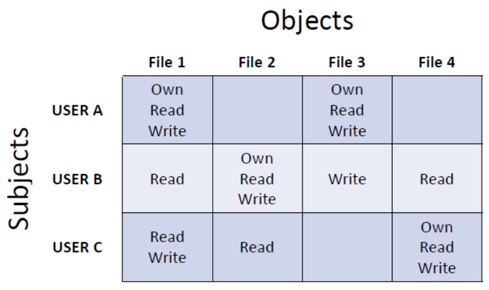
Access right: Describes the way in which a subject may access an object

- Read: view information in a system resource. Includes the ability to copy or print.
- Write: add, modify, or delete data in system resource. Includes read access.
- Execute: execute specified programs.
- Delete: delete certain system resources, such as files or records.
- Create: create new files, fields, etc.
- Search: list the files in a directory or otherwise search the directory.

4.2 Policies

4.2.1 Discretionary access control (DAC)

Scheme in which an entity may be granted access rights that permit the entity, by its own volition, to enable another entity to access some resource. Often provided using an access matrix where One dimension consists of identified subjects that may attempt data access to the resources and the other dimension lists the objects that may be accesse, each entry in the matrix indicates the access rights of a particular subject for a particular object.



Subjects	File 1	File 2	File 3	File 4
USER A	Own Read Write		Own Read Write	
USER B	Read	Own Read Write	Write	Read
USER C	Read Write	Read		Own Read Write

Objects

Figure 25: Access Matrix

4.2.1.1 Example of access control structures

ACL: Will be explanated after ...

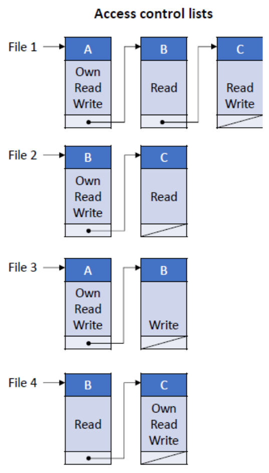


Figure 26: Access Control Lists

Lists of capabilities (CT) Specifies authorized objects and operations for a particular user. Each user has a number of tickets and may be authorized to loan or give them to others. However... tickets may be dispersed around the system, they present **a greater security** problem than ACL.

- The integrity of the ticket must be protected, and guaranteed. The ticket must be unforgeable.
- The OS may manage CT on behalf of users, but tickets would have to be held in a region of memory inaccessible to users.

- In alternative, they may include an unforgeable token in the capability (like a cryptographic message authentication code - MAC).
- The latter form of CT is appropriate in a distributed environment, when the security of its contents cannot be guaranteed.

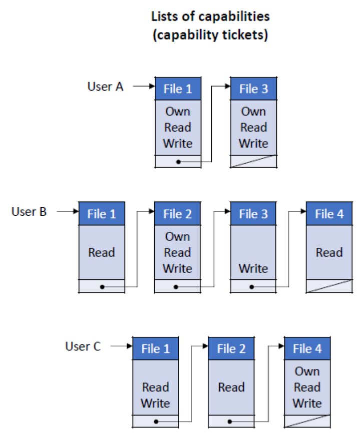


Figure 27: Lists of capabilities

Authorization Table Authorization table: more convenient than either ACLs or CT

- Sorted by subject gives CT
- Sorted by object gives ACL
- Easy to implement in a relational database

4.2.1.2 Models A model for DAC developed by Lampson, Graham, Denning (1971/72)

- Protection state of a system:
  - the set of information, at a given time, that specifies the access rights for each subject with respect to each object
- Requirements:
  - representing the protection state
  - enforcing access rights
  - allowing subjects to alter the protection state in certain ways

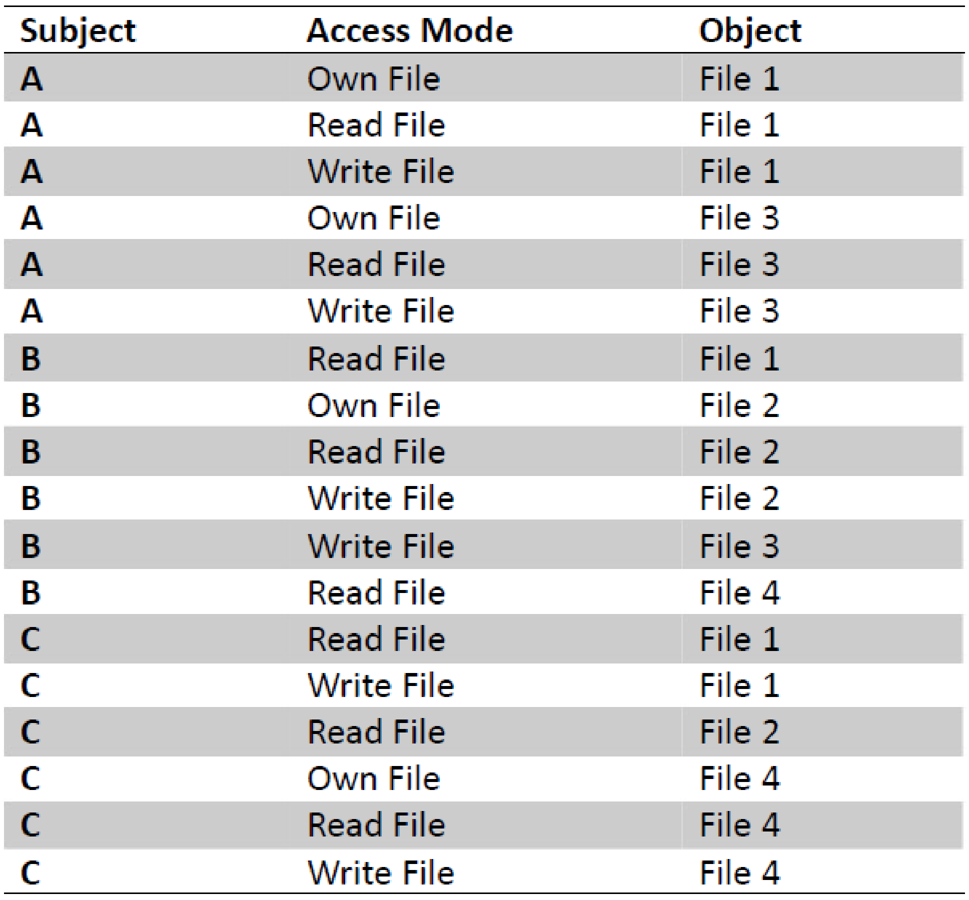
Representing the protection state Extends the universe of objects:

Processes: • Access rights include the ability to delete a process, stop (block), and wake up a process.

Devices: • Access rights include the ability to read/write the device, to control its operation (e.g., a disk seek), and to block/unblock the device for use.

Memory locations or regions: • Access rights include the ability to read/write certain regions of memory that are protected such that the default is to disallow access.

Subjects: • Access rights with respect to a subject have to do with the ability to grant or delete access rights of that subject to other objects, as explained subsequently.



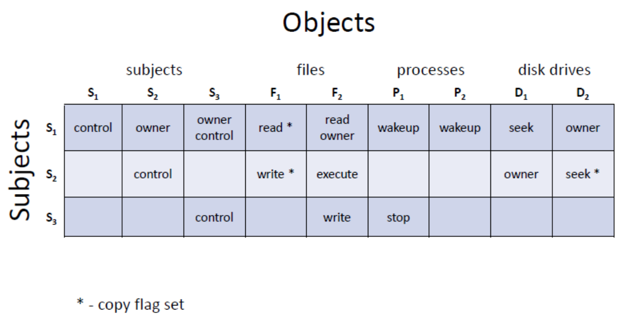
Subject	Access Mode	Object
A	Own File	File 1
A	Read File	File 1
A	Write File	File 1
A	Own File	File 3
A	Read File	File 3
A	Write File	File 3
B	Read File	File 1
B	Own File	File 2
B	Read File	File 2
B	Write File	File 2
B	Write File	File 3
B	Read File	File 4
C	Read File	File 1
C	Write File	File 1
C	Read File	File 2
C	Own File	File 4
C	Read File	File 4
C	Write File	File 4

Figure 28: Authorization Table

Enforcing Access Rights This model assumes a “separate” access control module for each object. An access triggers the following steps:

1. A subject S0 issues a request of type α for object X.
2. The request causes the system to generate a message of the form (S0, α, X) to the controller for X.
3. The controller interrogates the access matrix A to determine if α is in A[S0, X]:
   - If so, the access is allowed;
   - If not, the access is denied.

The model also includes rules for the modification of the access matrix, need for “owner” and “control” access rights to change the rights and the structure of the access matrix, and need for “copy flag” concept To give a right to another subject



Objects

		subjects			files		processes		disk drives	
		S1	S2	S3	F1	F2	P1	P2	D1	D2
Subjects	S1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S2		control		write *	execute			owner	seek *
	S3			control		write	stop			

* - copy flag set

Figure 29: Extended Access Control Matrix

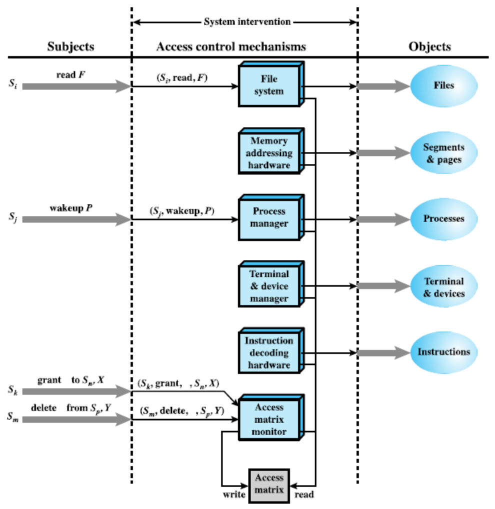
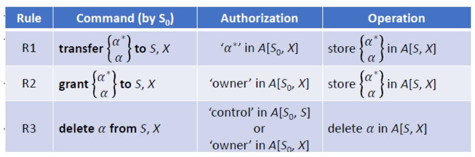


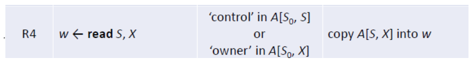
Figure 30: Organization of the access control function

4.2.1.3 Protection Domains So far, a subject is a user, and a row in the access control matrix represents its capabilities.It has more flexibility when associating capabilities with protection domains, permit to set of objects together with access rights to those objects, in terms of the access matrix, a row defines a protection domain. With flexibility a user can spawn processes with a subset of the access rights of the user and association between a process and a domain can be static or dynamic. In Unix, for example, in user mode certain areas of memory are protected from use and certain instructions may not be executed, while in kernel mode privileged instructions may be executed and protected areas of memory may be accessed



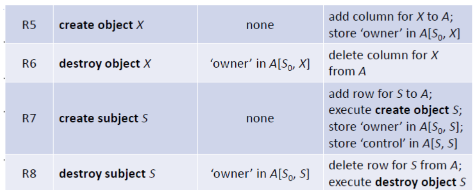
Rule	Command (by S_0)	Authorization	Operation
R1	**transfer** $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ **to** S, X	'α^*' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R2	**grant** $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ **to** S, X	'owner' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R3	**delete** α **from** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	delete α in $A[S, X]$

Figure 31: Transferring, granting and deleting access rights



R4	$w \leftarrow$ **read** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	copy $A[S, X]$ into w

Figure 32: Reading the access matrix



R5	**create object** X	none	add column for X to A; store 'owner' in $A[S_0, X]$
R6	**destroy object** X	'owner' in $A[S_0, X]$	delete column for X from A
R7	**create subject** S	none	add row for S to A; execute **create object** S; store 'owner' in $A[S_0, S]$; store 'control' in $A[S, S]$
R8	**destroy subject** S	'owner' in $A[S_0, S]$	delete row for S from A; execute **destroy object** S

Figure 33: Manipulating objects and subjects

Figure 34: Access control system commands

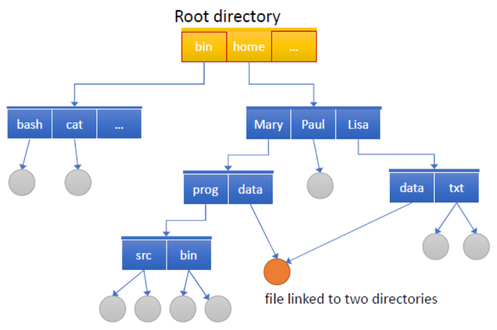


Figure 35: File System: tree structure

4.2.1.4 Recap of Operating Systems

Directories: In Unix each file (and directory) belongs to a directory and each directory is a data structure that links file names to file attributes an example of attributes are: file size, address on disk, access rights, time of last access, time of creation, ... A possible implementation is FAT file systems, where the directory is a table, it associates each file name to its file descriptor. In implementation of directories in Unix the directory is a table that includes the references to the file descriptors (i-nodes), which are stored in a separate data structure in the disk (which includes all attributes of the file)

/ it's the root directory

/bin binary executable files. These are utilities of the system

/boot everything required to boot the system

/dev contains the device files (¡¡everything is a file in Unix¿¿. . .)

- These are not regular files; accessing these files means accessing the respective system devices (drives, serial lines, etc. . .)

/etc contains system administration and configuration files (the file of passwords is here)

/home contains the users' home directories

/lib contains kernel modules and the shared libraries

/mnt generic point under which to mount a file system or a device

/opt contains add-on packages not part of the default installation

/proc is very special in that it is also a virtual filesystem (aka process information pseudo-file system):

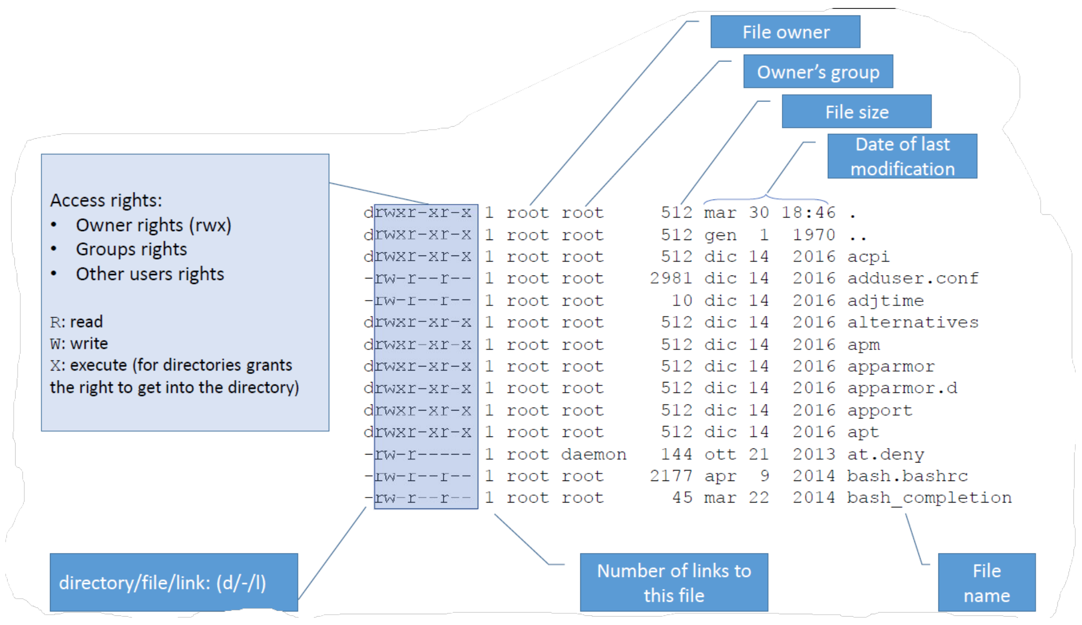
- It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc).
- It can be regarded as a control and information centre for the kernel.
- In fact, quite a lot of system utilities are simply calls to files in this directory.
- By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.

/tmp contains temporary files

/usr contains ¡¡everything userrelated¿¿, for example programs for users (not system programs).

/usr/bin programs for the user (included games)

/usr/lib libraries for user programs


File owner
Owner's group
File size
Date of last modification
Access rights:
• Owner rights (rwx)
• Groups rights
• Other users rights
R: read
W: write
X: execute (for directories grants the right to get into the directory)
directory/file/link: (d/-/l)
Number of links to this file
File name

```
drwxr-xr-x 1 root root     512 mar 30 18:46 .
drwxr-xr-x 1 root root     512 gen  1  1970 ..
drwxr-xr-x 1 root root     512 dic 14  2016 acpi
-rw-r--r-- 1 root root    2981 dic 14  2016 adduser.conf
-rw-r--r-- 1 root root      10 dic 14  2016 adjtime
drwxr-xr-x 1 root root     512 dic 14  2016 alternatives
drwxr-xr-x 1 root root     512 dic 14  2016 apm
drwxr-xr-x 1 root root     512 dic 14  2016 apparmor
drwxr-xr-x 1 root root     512 dic 14  2016 apparmor.d
drwxr-xr-x 1 root root     512 dic 14  2016 apport
drwxr-xr-x 1 root root     512 dic 14  2016 apt
-rw-r----- 1 root daemon   144 ott 21  2013 at.deny
-rw-r--r-- 1 root root    2177 apr  9  2014 bash.bashrc
-rw-r--r-- 1 root root      45 mar 22  2014 bash_completion
```

Figure 36: A directory in Unix

Access Right Creation of users and groups (and deletion) is reserved to the root user, by means of:

- `useradd` / `userdel`
- `groupadd` / `groupdel`

Changing Permissions to Files and Directories

To change permissions to files and directories:

- `chmod [a,u,g,o] [+,-] [r,w,x] file_name` :
  - gives (+) or revokes (-) rights [r,w,x] to all (a), owner (u), group (g), others (o) to the file `file_name`

Changing the Owner or the Group of a File

To change the owner or the group of a file:

- `chown new_owner file_name` (change owner)
- `chgrp new_group file_name` (change group)

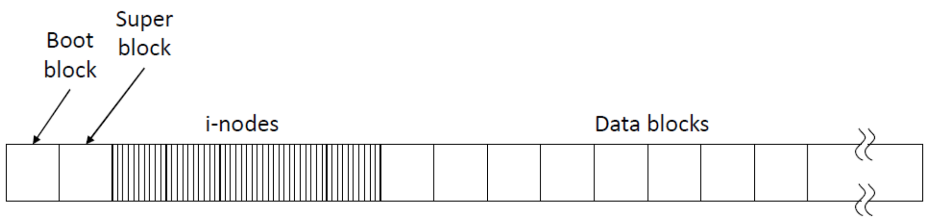


Figure 37: Physical disk organization in UNIX

Berkeley UNIX FFS (Fast File System) In this kynd of file system there is a *inode table* that contains all inodes, that are one per file, contains metadata (like file owner, access permissions, access times ...) and a set of pointers to data blocks.

4.2.1.5 Unix File access controll

Files UNIX files are administered using inodes (index nodes), which are control structures containing key information needed for a particular file. Several file names may be associated with a single inode. An active inode is associated with exactly one file. File attributes, permissions, and control information are stored in the inode. On the disk, there is an inode table, or inode list, that contains the inodes of all the files in the file system. When a file is opened, its inode is brought into main memory and stored in a memory resident inode table.

Directories May contain files and/or other directories and contains file names plus pointers to associated inodes

Traditional

- A user:
  - Unique user identification number (user ID)
  - Member of a primary group identified by a group ID
  - May also belong to several other groups
- A file (an object):
  - Has an owner
  - 12 protection bits
  - Specify read, write, and execute permission for the owner of the file, members of the group and all other users
  - The owner ID, group ID, and protection bits are part of the file's inode
- A hierarchy:
  - owner – group – all others
- Concerning directories:
  - Read grants the list capability
  - Write grants the create / rename / delete files capability
  - Execute grants the ability to descend into subdirectories
- "Set user ID" (SetUID) and "Set group ID" (SetGID)
  - System temporarily uses rights of the file owner/group in addition to the real user's rights when making access control decisions
  - Enables privileged programs to access files/resources not generally accessible
- Sticky bit
  - When applied to a directory it specifies that only the owner of any file in the directory can rename, move, or delete that file

- Superuser
  - A specific User ID
  - Is exempt from usual access control restrictions
  - Has system-wide access

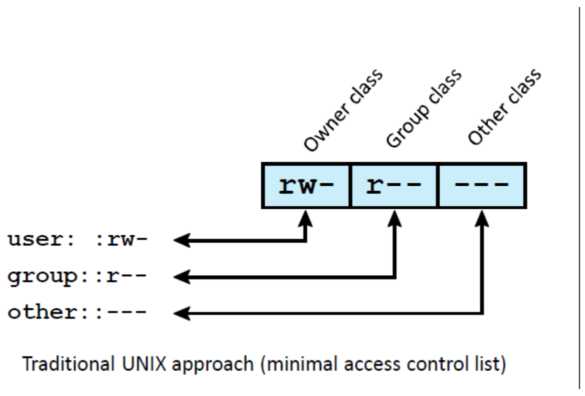


Figure 38: Traditiona ACL

Modern Many moderna Unix systems support ACLs as OpenBSD,Linux,Solaris, FreeBSD. In the last The administrator can assign a list of UNIX user IDs and groups by means of the Setfacl command. Any number of users and groups can be associated with a file, each with read, write, and execute protection bits. A file does not need to have an ACL; it includes an additional protection bit that indicates whether the file has an extended ACL.

Steps for process request:

1. Select the ACL that closely matches the requesting process
   - In this order: owner, named users, groups, others
2. Check if the matching entry contains sufficient permissions
   - A process can be in more than one group, just one is sufficient

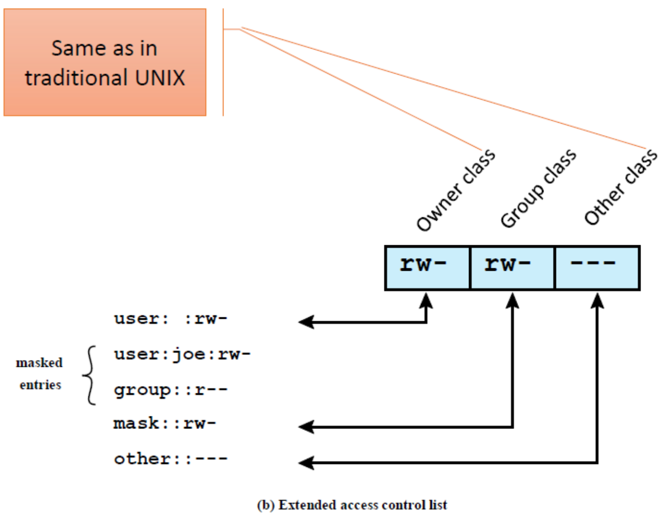
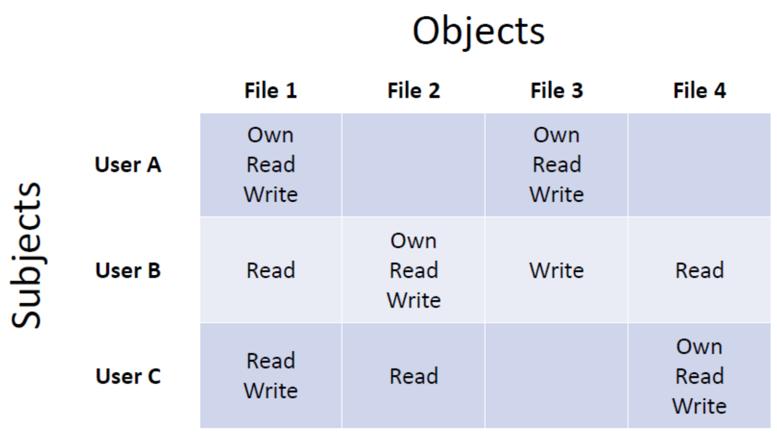


Figure 39: Traditiona ACL

4.2.1.6 Exercises

1. For the DAC model, an alternative representation of the protection state is a directed graph.
2. Each subject and each object in the protection state is represented by a node (a single node is used for an entity that is both subject and object).
3. A directed line from a subject to an object indicates an access right, and the label on the link defines the access right.

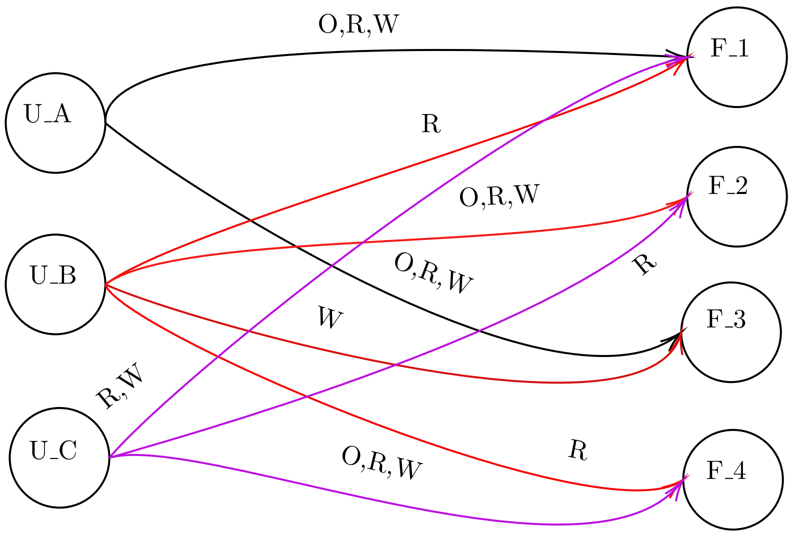
Question Draw a directed graph that corresponds to the access matrix of the figure



Objects

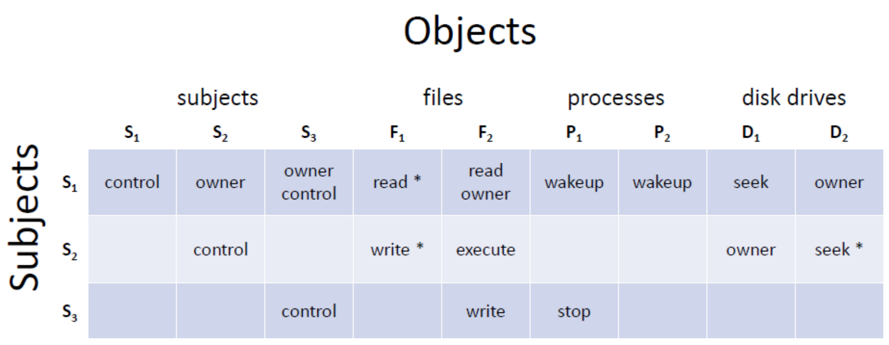
Subjects

	File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write



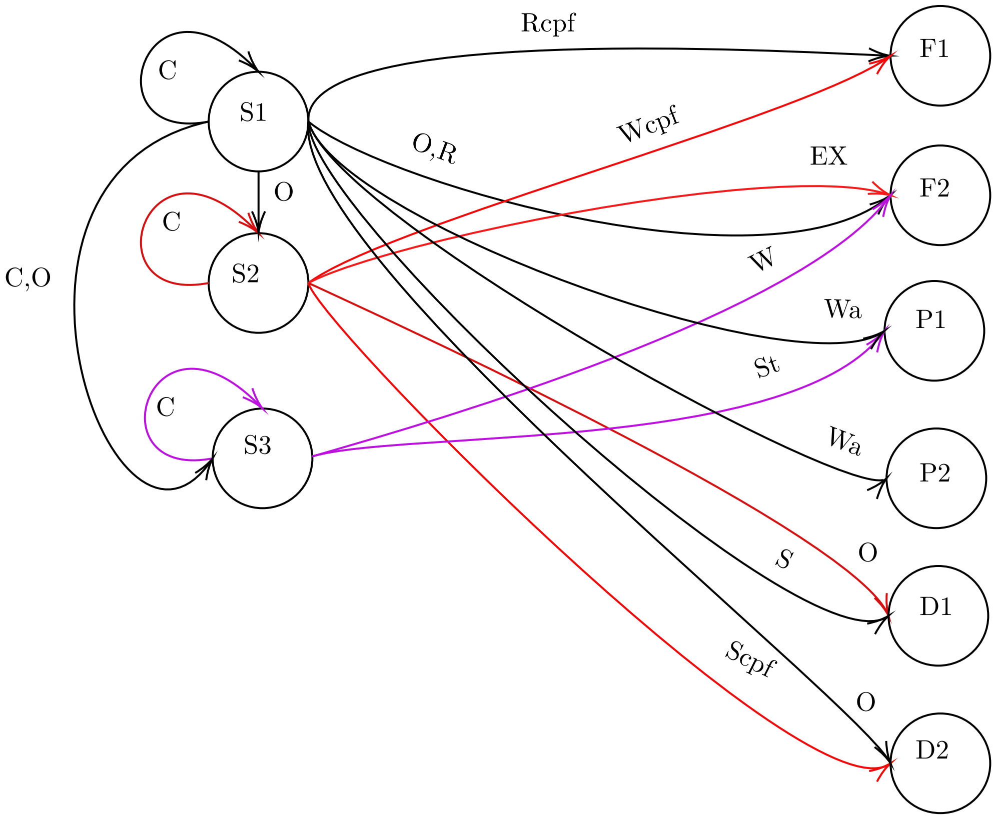
Answer

Question Draw a directed graph that corresponds to the access matrix of the table:



Objects

Subjects	subjects			files		processes		disk drives	
	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
S_2		control		write *	execute			owner	seek *
S_3			control		write	stop			



Answer

Question Is there a one-to-one correspondence between the directed graph representation and the access matrix representation? Explain.

Answer There is indeed a one-to-one correspondence between the directed graph representation and the access matrix representation of access control models. Here is how they correspond:

1. **Access Matrix Representation:**
   - The access matrix is a two-dimensional matrix where rows represent subjects (users or processes), and columns represent objects (files or resources).
   - Each entry in the matrix specifies the access rights that a particular subject has on a particular object. The rights can include permissions such as read, write, and execute.

2. **Directed Graph Representation:**
   - In the directed graph representation, each subject and object is represented as nodes in the graph.
   - Directed edges (or arcs) from a subject node to an object node represent access rights. Each edge is labeled with the specific access right (e.g., read, write) granted from the subject to the object.

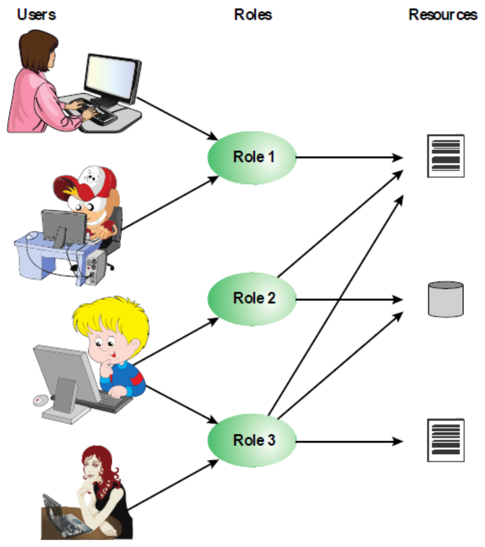
3. **Correspondence:**

- Each entry in the access matrix corresponds to a directed edge in the graph. Specifically, if a subject S_i has access right R to an object O_j in the access matrix, then there is a directed edge from the node representing S_i to the node representing O_j in the graph, labeled with the right R.
- Conversely, each directed edge in the graph corresponds to an entry in the access matrix. If there is an edge from node S_i to node O_j labeled with right R, the entry in the access matrix at row i and column j will indicate that subject S_i has access right R on object O_j.

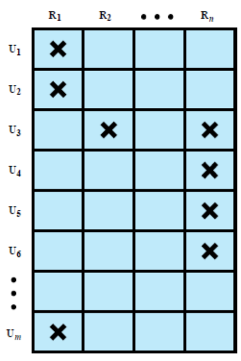
Thus, the directed graph and the access matrix representations are two different ways of expressing the same information about access rights in a system, and there is a one-to-one correspondence between them.

4.2.2 Role Based Access Control

The transition from users' identities to roles involves defining job functions within an organization. Rights are assigned to these roles, allowing users to change roles and thereby alter their rights. This approach has seen widespread commercial use and was formalized as a NIST standard in 2009.

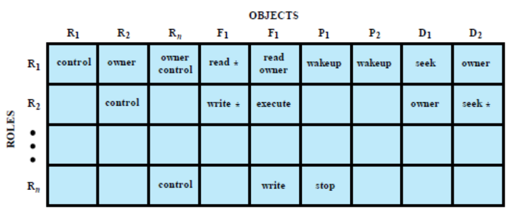


Relation users-roles



	R_1	R_2	• • •	R_n
U_1	✖			
U_2	✖			
U_3		✖		✖
U_4				✖
U_5				✖
U_6				✖
⋮				
U_m	✖			

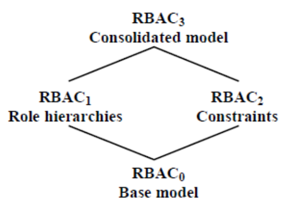
Access control matrix (roles/objects)



OBJECTS

ROLES	R_1	R_2	R_n	F_1	F_1	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R_2		control		write *	execute			owner	seek *
⋮									
R_n			control		write	stop			

Figure 40: Access control matrix representation of RBAC

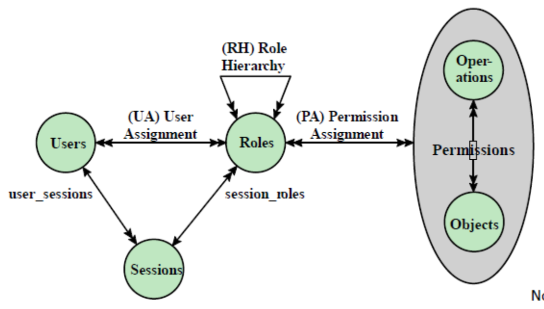


Four models related to each other

Models	Hierarchies	Constraints
RBAC$_0$	No	No
RBAC$_1$	Yes	No
RBAC$_2$	No	Yes
RBAC$_3$	Yes	Yes

Scope of RBAC models

4.2.2.1 A family of rolebased access control models



RBAC$_0$ model

Note:
single arrow means one,
double arrow means many.

RBAC0

- RBAC0 does not have role hierarchy and constraints
- It contains the four types of entities:
  - **User**: An individual (with user ID) that has access to this computer system.
  - **Role**: A named job function within the organization that controls this computer system (includes a description of the authority and responsibility conferred on this role).
  - **Permission**: An approval of a particular mode of access to one or more objects (i.e., access right, privilege, and authorization).
  - **Session**: A mapping between a user and an activated subset of the set of roles to which the user is assigned.

RBAC1

- Role hierarchies provide a means of reflecting the hierarchical structure of roles in an organization
  - Greater responsibility implies greater authority to access resources
  - Inheritance of access rights from the bottom to the top

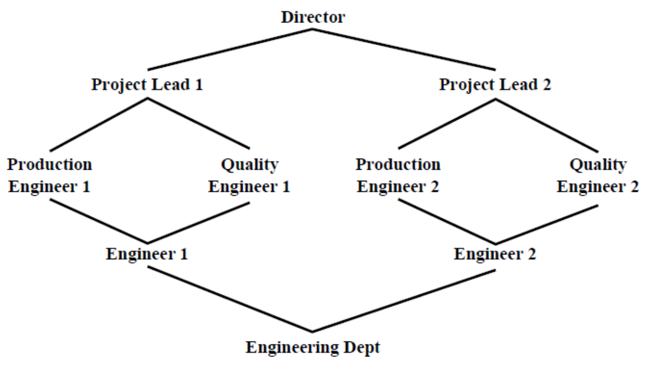


Figure 41: Example of ROI hierarchy

RBAC2 Provide a means of adapting RBAC to the specifics of administrative and security policies of an organization a constraint is a relationship among roles or a condition related to roles.

Types of constraints:

- Mutually exclusive roles
  - A user can only be assigned to one role in the set (either during a session or statically)
  - Any permission (access right) can be granted to only one role in the set
- Cardinality
  - Setting a maximum number of users in a role
  - May also limit the number of roles per user, or the number of roles a user can have per session
- Prerequisite roles
  - Dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role

4.2.3 Attribute-Based Access Control

The authorization model can define authorizations that express conditions on properties of both the resource and the subject. The strength of this model lies in its flexibility and expressive power. However, the main obstacle to its adoption in real systems has been the concern about the performance impact of evaluating predicates on both resource and user properties for each access.

Web services have been pioneering technologies through the introduction of the eXtensible Access Control Markup Language (XACML). There is considerable interest in applying this model to cloud services.

ABAC is distinguishable because it controls access to objects by evaluating rules against the attributes of entities, operations, and the environment relevant to a request. It relies on the evaluation of attributes of the subject, attributes of the object, and a formal relationship or access control rule defining the allowable operations for subject-object attribute combinations in a given environment. ABAC systems are capable of enforcing DAC, RBAC, and MAC concepts. Additionally, ABAC allows an unlimited number of attributes to be combined to satisfy any access control rule. Three key elements of an ABAC system:

- **Attributes** – defined for entities in a configuration;
- **Policy Model** – defines the ABAC policies;
- **Architecture Model** – applies to policies that enforce access control.

4.2.3.1 Attribute

Subject Attributes A subject is an active entity that causes information to flow among objects or changes the system state and attributes define the identity and characteristics of the subject (id, name, job title, etc.)

Object attributes An object (or resource) is a passive information systemrelated entity containing or receiving information, it have attributes that can be leverages to make access control decisions, and its attribute depend on the object... (owner, author, date, QoS,...)

environment attributes The operational, technical, and situational environment or context in which information access occurs is described through contextual attributes. These attributes have largely been ignored in most access control policies. Examples of such attributes include current time, virus/hacker activity, and network security level.

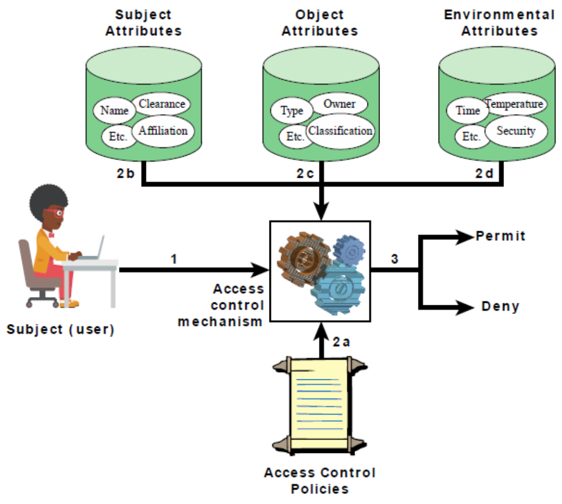


Figure 42: ABAC Scenario

4.2.3.2 Policies

A policy is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under specific environmental conditions. Policies are typically written from the perspective of the object that needs protection and the privileges available to subjects. Privileges represent the authorized behavior of a subject, are defined by an authority, and embodied in a policy. Other terms commonly used instead of privileges are rights, authorizations, and entitlements.

Notations

1. S, O, and E are subjects, objects, and environments, respectively;
2. SA_k ($1 \leq k \leq K$): predefined attributes for subjects;
   OA_m ($1 \leq m \leq M$): predefined attributes for objects;
   EA_n ($1 \leq n \leq N$): predefined attributes for environments;
3. ATTR(s), ATTR(o), and ATTR(e) are attribute assignment relations for subject s, object o, and environment e, respectively:
   - $ATTR(s) \subseteq SA_1 \times SA_2 \times ... \times SA_K$
   - $ATTR(o) \subseteq OA_1 \times OA_2 \times ... \times OA_M$
   - $ATTR(e) \subseteq EA_1 \times EA_2 \times ... \times EA_N$

   Examples of attributes:
   - Role(s) = “Service Consumer”
   - ServiceOwner(o) = “XYZ, Inc.”
   - CurrentDate(e) = “01-23-2005”
4. A Policy Rule:
   - Decides on whether a subject s can access an object o in a particular environment e;
   - It is a Boolean function of the attributes of s, o, and e:

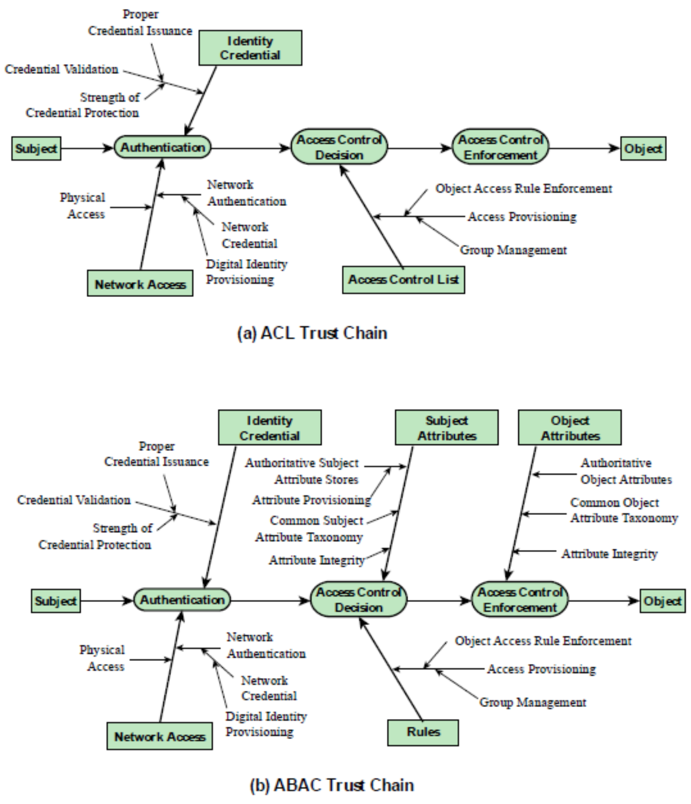


Figure 43: ACL and ABAC relationships

- Rule: can_access(s, o, e) ← f(ATTR(s), ATTR(o), ATTR(e))

5. A Policy Rule Base (or policy store):
   - May consist of a number of policy rules;
   - Each rule covers many subjects and objects within a security domain;
   - The access control decision process in essence amounts to the evaluation of applicable policy rules in the policy store.

4.2.3.3 RBAC vs ABAC Assume in RBAC, if there are K subject attributes and M object attributes:

- the number of required roles are: $\prod_{k=1}^{K} range(SA_k)$
- the number of required permission are : $\prod_{m=1}^{M} range(SA_m)$

where range(·) denotes the number of possible values of each attribute In contrast ABAC is more efficient, it is just sufficient to add an attribute.

4.2.3.4 Exercises

Question Now consider the example of an online entertainment store that streams movies to users for a flat monthly fee. The store must enforce the following access control policy based on the user's age and the movie's content rating:

Movie	Rating	Users Allowed Access
R	Age 17 and older	
PG-13	Age 13 and older	
G	Everyone	

Table 7: Movie Ratings and Allowed Access

Model the access control po licies with RBAC and with ABAC

- RBAC To do:
  1. Define the roles (?)
  2. Define the permissions (?)
  3. Define which permissions are given to each role (?)
- ABAC To do:
  1. Define the roles (?)
  2. Define attributes (?)
  3. Define policy rules (?)

Answer RBAC:
We can define 3 roles (R):

R1 : Adult ≥ 17

R2 : Young people $\geq 13 \wedge < 17$

R3 : Child < 13

And then we can define 3 permissions (P) based on rate:

P1 : Can view R movie

P2 : Can view PG-13 movie

P3 : Can view G movie

	P1	P2	P3
R1	X	X	X
R2		X	X
R3			X

ABAC We can define a user u that can access of view a movie m (the security environment e is ignored here). We can resolve with this policy rule:

$$R1 : Can\_Access(u, m, e) \leftarrow (Age(u) \geq 17 \wedge Rating(m) \in \{R, PG-13, G\})$$
$$\vee(Age(u) \geq 13 \ \wedge Age(u) < 17 \wedge Rating(m) \in \{PG-13, G\}) \vee (Age(u) < 13 \wedge Rating(m) \in \{G\})$$

Question Assume now that:

- movies can also be classified as "New Release" or "Old Release"
- Users are also classified as "Premium user" or " Regular User"

Model RBAC and ABAC so that only Premium Users can see the New Releases.

Answer **RBAC:** if we add two new categories, the number of rule become double, because 3 categories on the age and 2 on type of user, we have 3*2 role

ABAC We can add 2 policies for resolve this:

$$R2 : Can\_Access(u, m, e) \leftarrow (MemberType(u) = Regular \lor MemberType(u) = Premium \land MovieRelease(m) = Old)$$
$$\lor(MemberType(u) = Premium \land MovieRelease(m) = Old)$$
$$R3 : Can\_Access(u, m, e) \leftarrow (R1 \land R2)$$

4.3 Identity, credential, and access management (ICAM) and trust frameworks

ICAM is a comprehensive approach to managing and implementing digital identities, credentials, and access control. It was developed by the U.S. government and is designed to:

- Create trusted digital identity representations of individuals and nonperson entities (NPEs).
- Bind those identities to credentials that may serve as a proxy for the individual or NPE in access transactions.
- Use the credentials to provide authorized access to an agency's resources.

A credential is an object or data structure that authoritatively binds an identity to a token possessed and controlled by a subscriber.

4.3.1 Identity Management

Identity management involves assigning attributes to a digital identity and linking it to an individual or nonperson entity (NPE), with the goal of establishing a trustworthy digital identity that is application- and context-independent. The common approach to access control is to create a digital identity for specific applications or programs, but maintaining and protecting the identity itself is often secondary to the application's mission. Unlike accounts, digital identity is not tied to job titles, duties, locations, or other attributes; these are merely attributes connected to the identity. Access control is thus based on the context and relevant user attributes rather than solely on their identity. This allows individuals to have a single digital representation that can be leveraged across departments and agencies for various purposes, including access control.

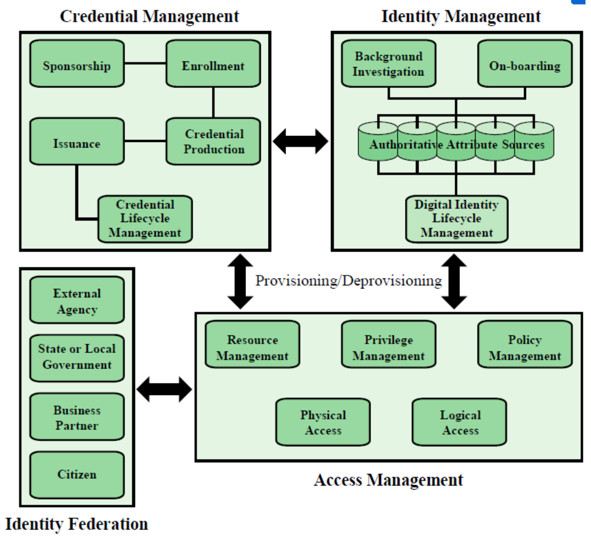


Figure 44: Key functions in identity management

The final element is lifecycle management, which includes:

- Mechanisms, policies, and procedures for protecting personal identity information.
- Controlling access to identity data.
- Techniques for sharing authoritative identity data with applications that need it.
- Revocation of an enterprise identity.

4.3.2 Credential Management

A credential is an object or data structure that authoritatively binds an identity to a token possessed and controlled by a subscriber. Examples of credentials are smart cards, private/public cryptographic keys, and digital certificates. The management of the life cycle of the credential encompasses five logical components:

1. An authorized individual sponsors an individual or entity for a credential to establish the need for the credential.
2. The sponsored individual enrolls for the credential.
   - Process typically consists of identity proofing and the capture of biographic and biometric data.
   - This step may also involve incorporating authoritative attribute data, maintained by the identity management component.
3. A credential is produced.
   - Depending on the credential type, production may involve encryption, the use of a digital signature, the production of a smart card, or other functions.
4. The credential is issued to the individual or NPE.
5. A credential must be maintained over its life cycle.
   - Might include revocation, reissuance/replacement, reenrollment, expiration, personal identification number (PIN) reset, suspension, or reinstatement.

4.3.3 Access Management

Access management and control deals with the ways entities are granted access to resources, covering both logical and physical access. This can be internal to a system or involve external elements. The purpose is to ensure proper identity verification when an individual attempts to access a security-sensitive building, computer systems, or data.

4.3.3.1 Three support elements are needed for an enterprise-wide access control facility:

Resource management Access control is concerned with defining rules for resources that require controlled access. These rules specify credential requirements and outline the necessary user attributes, resource attributes, and environmental conditions needed to access a given resource for a particular function.

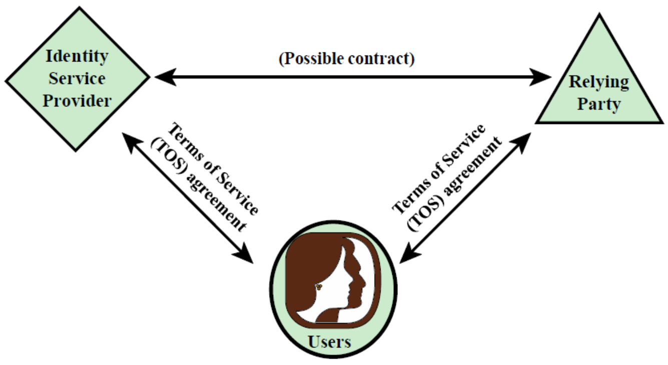
Privilege management Entitlement and privilege management is concerned with establishing and maintaining the attributes that comprise an individual's access profile. These attributes represent features of an individual used to determine access decisions to both physical and logical resources. Privileges are considered attributes that can be linked to a digital identity.

Policy management Governs what is allowable and unallowable in an access transaction

4.3.4 Identity federation

Term used to describe the technology, standards, policies, and processes that allow an organization to trust digital identities, identity attributes, and credentials created and issued by another organization.
Addresses two questions:

4.3.4.1 Trust How do you trust identities of individuals from external organizations who need access to your systems?
Trust, identity, and attributes have become primary concerns in internet business, network service providers, and similar fields. Consider the case of e-commerce: the attributes you need to know about someone to deal with them depend on the specific situation. These attributes might include professional registration or license number, organization and department, staff ID, security clearance, customer reference number, credit card number, unique health identifier, allergies, blood type, Social Security number, address, citizenship status, social networking handle, pseudonym, and more. The attributes that must be known and verified to permit a transaction are context-dependent. This issue extends beyond just e-commerce.


Identity Service Provider
(Possible contract)
Relying Party
Terms of Service (TOS) agreement
Terms of Service (TOS) agreement
Users

Traditional triangle of parties involved in an exchange of identity information

4.3.4.2 Identity Information exchange approaches How do you vouch for identities of individuals in your organization when they need to collaborate with external organizations?

Relying party:
- requires that the user has been authenticated to some degree of assurance,
- attributes provided by the identity service provider are accurate,
- identity service provider is authoritative for those attributes.

Identity service provider:
- requires assurance of accurate user information,
- if sharing information, the relying party will use it according to contractual terms and the law.

User:
- requires assurance that the identity service provider and relying party can be trusted with sensitive information,
- that they will respect user preferences and privacy.

All parties:
- want to ensure that the practices described by other parties are actually implemented and reliable.

Open Identity Trust framework A trust framework functions as a certification program, enabling a party who accepts a digital identity credential (the relying party) to trust the identity, security, and privacy policies of the party who issues the credential (the identity service provider) and vice versa. Developed by a community with similar goals and perspectives, a trust framework defines the rights and responsibilities of the community's participants, specifies policies and standards, and outlines the processes and procedures that provide assurance. Different trust frameworks can exist, allowing participants to tailor them to meet their specific needs.

OpenID: An open standard that allows users to be authenticated by certain cooperating sites using a third party service.

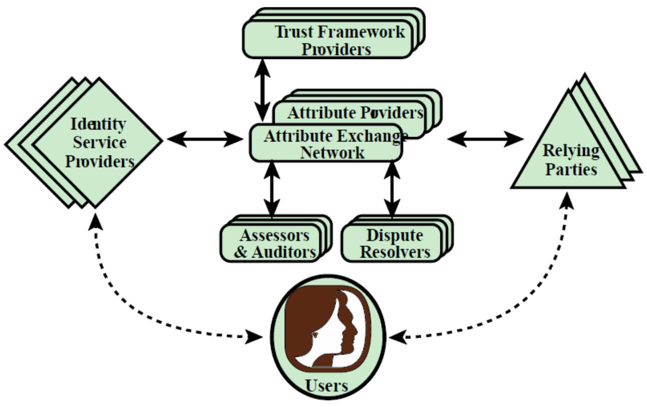
OIDF: OpenID Foundation is an international nonprofit organization of individuals and companies committed to enabling, promoting, and protecting OpenID technologies.

ICF: Information Card Foundation is a nonprofit community of companies and individuals working together to evolve the Information Card ecosystem.

OITF: Open Identity Trust Framework is a standardized, open specification of a trust framework for identity and attribute exchange, developed jointly by OIDF and ICF.

OIX: Open Identity Exchange Corporation is an independent, neutral, international provider of certification trust frameworks conforming to the OITF model.

AXN: Attribute Exchange Network is an online Internet-scale gateway for identity service providers and relying parties to efficiently access user asserted, permissioned, and verified online identity attributes in high volumes at affordable costs.


Trust Framework Providers
Attribute Providers
Attribute Exchange Network
Identity Service Providers
Relying Parties
Assessors & Auditors
Dispute Resolvers
Users

Identity attribute exchange elements

5 Database and Data Center Security Computer

Reasons database security has not kept pace with the increased reliance on databases are:

- There is a dramatic imbalance between the complexity of modern database management systems (DBMS) and the security techniques used to protect them.
- Databases have sophisticated interactions. The Structured Query Language (SQL) protocol is complex.
- There is an increasing reliance on cloud technology to host part or all of the corporate database.
- Effective database security requires a strategy based on a full understanding of the security vulnerabilities of SQL.
- The typical organization lacks full-time database security personnel.
- Most enterprise environments consist of a heterogeneous mixture of database platforms, enterprise platforms, and OS platforms, creating an additional complexity hurdle for security personnel.

5.1 DBMS

5.1.0.1 Databases A database is a structured collection of data stored for use by one or more applications. It contains the relationships between data items and groups of data items, and it can sometimes include sensitive data that requires security measures. A query language provides a uniform interface to the database for both users and applications.

5.1.0.2 DMBS A database management system (DBMS) is a suite of programs designed for constructing and maintaining databases. It offers ad hoc query facilities to multiple users and applications, allowing flexible and efficient access to the stored data.

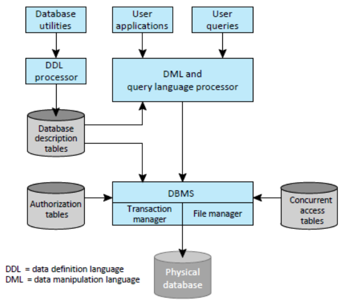


Figure 45: DBMS architecture

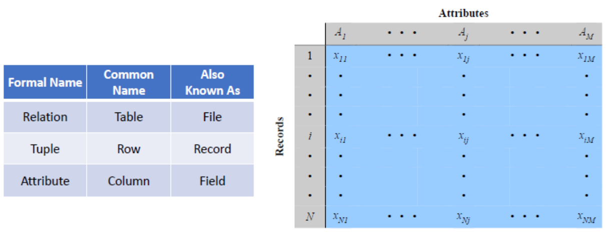
Security Requirements Database systems generate security requirements that exceed the capabilities of typical OS-based security mechanisms or stand-alone security packages. While operating system security mechanisms generally control read and write access to entire files, they are inadequate for limiting access to specific records or fields within those files. DBMSs, however, require such detailed (granular) access control and typically enable access controls over a broader range of commands, such as selecting, inserting, updating, or deleting specified items in the database. This necessitates specifically designed security services and mechanisms integrated with the DBMS.

5.1.0.3 Relational Database

A table in a database consists of rows and columns, where each column holds a particular type of data and each row contains specific values for each column. Ideally, one column contains unique values, serving as an identifier or key for that row. This structure allows for the creation of multiple tables linked by a unique identifier present in all tables. A relational query language is used to access the database, enabling users to request data that meet specific criteria. This query language is declarative, meaning it specifies what data to retrieve rather than how to retrieve it.

Elements:

- **Relation**: A flat table in a file.
- **Tuple**: Rows (records) of the table.
- **Attribute**: Columns (fields) of the table.
- **Primary key**: Uniquely identifies a row; consists of one or more column names.
- **Foreign key**: Links one table to attributes in another; it's a primary key in another table.
- **View/virtual table**: The result of a query that returns selected rows and columns from one or more tables. Views are often used for security purposes to provide restricted access to certain rows or columns.


Attributes
A_1 … A_j … A_M
Records
1
i
N
x_{11} … x_{1j} … x_{1M}
x_{i1} … x_{ij} … x_{iM}
x_{N1} … x_{Nj} … x_{NM}

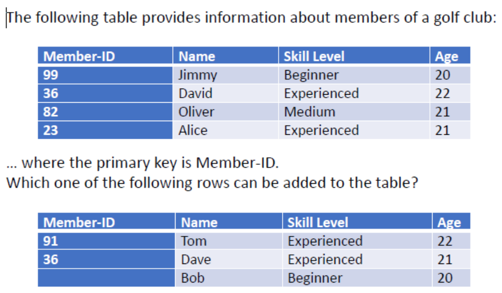
Formal Name	Common Name	Also Known As
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Figure 46: Basic Terminolgy

5.1.0.4 SQL

SQL is a standardized language used to define schemas, manipulate data, and query information in a relational database. While there are several similar versions of the ANSI/ISO standard, all adhere to the same basic syntax and semantics. SQL statements can be employed to create tables, insert and delete data within those tables, create views, and retrieve data through query statements.

5.1.0.5 Question & Examples



The following table provides information about members of a golf club:

Member-ID	Name	Skill Level	Age
99	Jimmy	Beginner	20
36	David	Experienced	22
82	Oliver	Medium	21
23	Alice	Experienced	21

... where the primary key is Member-ID.
Which one of the following rows can be added to the table?

Member-ID	Name	Skill Level	Age
91	Tom	Experienced	22
36	Dave	Experienced	21
	Bob	Beginner	20

Question

Answer Only rows with Member-ID equals to 91, 36 can be added to the table

5.2 SQL injection attacks

SQL injection is one of the most prevalent and dangerous network-based security threats, where malicious SQL commands are sent to a database server. The most common goal of such attacks is the bulk extraction of data. However, depending on the environment, SQL injection can also be exploited to modify or delete data, execute arbitrary operating system commands, and launch denial-of-service (DoS) attacks.

5.2.1 SQL Injection Attacks

An SQL injection (SQLi) attack is designed to exploit the dynamic nature of modern web application pages, which are no longer static and often query databases on servers to access sensitive data. By sending malicious SQL commands to the database server, an SQLi attack can extract large amounts of data, modify or delete data, execute arbitrary operating system commands, and even launch denial-of-service (DoS) attacks.

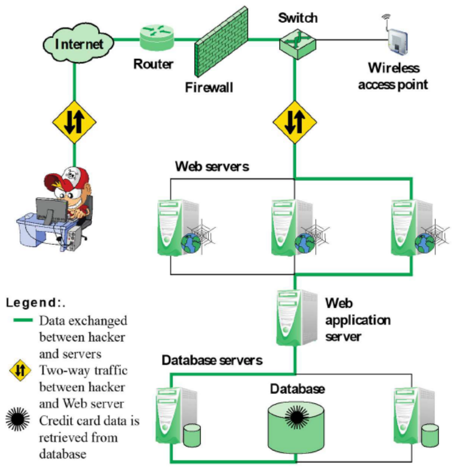

Internet
Router
Firewall
Switch
Wireless access point
Web servers
Web application server
Database servers
Database
Legend:.
Data exchanged between hacker and servers
Two-way traffic between hacker and Web server
Credit card data is retrieved from database

Figure 47: A typical SQL injection attack

5.2.1.1 Injection Technique The SQLi attack typically works by prematurely terminating a text string and appending a new command, because the inserted command may have additional strings appended to it before it is executed the attacker terminates the injected string with a comment mark ”- -”, subsequent text is ignored at execution time

5.2.1.2 Example of SQLi As a simple example, consider a script that builds an SQL query by combining predefined strings with text entered by a user:

```
var ShipCity;
ShipCity = Request.form ("ShipCity");
var sql ="SELECT * FROM OrdersTable WHERE ShipCity = '" + ShipCity +"
```

The intention of the script's designer is that a user will enter the name of a city. For example, if the user enters Pisa, then the following SQL query is generated:

```
SELECT * FROM OrdersTable WHERE ShipCity = 'Pi  sa';
```