



POLITECNICO DI TORINO

MASTER DEGREE IN COMPUTER ENGINEERING

Information Systems Security

NOTES FROM THE COURSE 01TYMUV OF PROF. ANTONIO LIOY
A.A. 2023/24

Main authors:

Marco Smorti
Riccardo Zaccone
Giovanni Nicosia

IMPORTANT NOTICE: these notes are not in any way promoted or checked by the professor or any person of the course teaching staff, so they are provided “AS-IS” and without any guarantee. However, we put a great effort to make them and verify their correctness, we hope they will be useful for you.

Version 2.0
October 10, 2023

Preliminary notes

Copyright (C) 2021.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the appendix A entitled "GNU Free Documentation License".

Contributors

- Marco Smorti
- Riccardo Zaccone
- Giovanni Nicosia

Contents

1	Introduction to the security of ICT systems	6
1.1	Why is security an important issue?	6
1.2	Complexity of the ICT scenario	7
1.3	A definition of ICT Security [OLD]	7
1.4	Risk estimation	9
1.5	Risk management	10
1.6	Analysis and management of security	10
1.7	Relation in the security field	11
1.8	Relation in the security field	11
1.9	Window of exposure	13
1.10	Some statistics	13
1.11	Cyber threats	15
1.12	What is Security?	16
1.13	Security properties	17
1.13.1	Peer authentication	18
1.13.2	Non-repudiation	19
1.13.3	Availability	19
1.13.4	Authorization (access control)	19
1.13.5	Privacy	20
1.13.6	Integrity	21
1.14	Data Protection	22
1.15	Some classes of attacks	23
1.15.1	Packet Sniffing (Eavesdropping)	24
1.15.2	Traffic Analysis	24
1.15.3	IP Spoofing (Masquerading)	24
1.15.4	Denial-of-Service (DoS)	25
1.15.5	Distributed Denial-of-Service (DDoS)	25
1.15.6	Shadow Server / Fake Server	27
1.15.7	Connection Hijacking / Man In The Middle (MITM)	28
1.15.8	Trojan	28
1.15.9	Zeus	28
1.15.10	Software Bug	29

1.16 Virus & Co. (malware)	29
1.17 Basic problems (non-technological)	33
1.18 Some important recent attacks	37
1.18.1 Against Cyber-Physical Systems	37
1.18.2 Against critical infrastructure	39
1.19 Attack maps: sample data sources	40
1.20 The three (four) pillars of security	41
1.21 The NIST cybersecurity framework	42
1.22 Final slides	42
2 Cryptographic techniques for cybersecurity	43
2.1 Cryptography	43
2.2 Symmetric and asymmetric cryptography	44
2.2.1 Application of block algorithm	48
3 Authentication techniques and architectures	55
3.1 What is authentication	55
3.2 Digital authentication model (NIST SP800.63B)	56
3.3 Generic authentication protocol	58
3.4 Passwords (reusable)	58
3.4.1 The "Dictionary" Attack	61
3.4.2 Rainbow table	62
3.4.3 Using salt in storing passwords	63
3.5 Strong (peer) authN	65
3.5.1 ECB Definition for Internet Banking	65
3.5.2 PCI-DSS definition for payment with credit cards	65
3.5.3 PCI-DSS Definition for Payment with Credit Cards	66
3.5.4 Other definitions	66
3.6 Challenge-Response Authentication (CRA)	66
4 Security of IP networks	68
5 Security of network applications	69
6 Firewall and IDS/ISP	70
7 E-mail security	71
A GNU Free Documentation License	72
GNU Free Documentation License	72
1. APPLICABILITY AND DEFINITIONS	72
2. VERBATIM COPYING	74
3. COPYING IN QUANTITY	74
4. MODIFICATIONS	75

5. COMBINING DOCUMENTS	76
6. COLLECTIONS OF DOCUMENTS	77
7. AGGREGATION WITH INDEPENDENT WORKS	77
8. TRANSLATION	77
9. TERMINATION	78
10. FUTURE REVISIONS OF THIS LICENSE	78
11. RELICENSING	78

Chapter 1

Introduction to the security of ICT systems

1.1 Why is security an important issue?

Cybersecurity has become very important in today's world. Since every system relies on computer systems, any kind of damage can result in significant economic losses. Even indirect attacks that do not aim to steal money have economic costs. Cybersecurity is essential because attacks can be performed without the need to physically access the target location.

The reasons why cybersecurity is important are as follows:

- Big damage on successful attacks
- Easy accessibility of systems

We must consider all the possible consequences of a successful attack. First, there can be **financial loss** (direct loss, for example, if someone gains access to bank account credentials, and indirect loss if the revelation that the company has been attacked negatively affects the stock exchange). There can be **recovery costs** because every successful attack results in damage, and there will be expenses required to return the system to normal operations and to enhance it to prevent new attacks. There can also be **productivity losses** if the attacks halt or delay processes. A successful attack may lead to **business disruption** because customers may seek alternative suppliers if a company is vulnerable to attacks.

For all these reasons we should protect systems. Most of the innovation is based on two main pillars:

- The ability to communicate from any part of the world (communication networks)
- The increasing use of personal and mobile devices

These two foundations are no longer sufficient for innovative products; every new product now requires a security system.

1.2 Complexity of the ICT scenario

The ICT scenario is complex for various reasons. One key factor is the sheer number of different mobile and connected **devices**, including desktops, laptops, tablets, smartphones, smart TVs, fridges, and cars. All these devices can now connect to the internet, making security a critical concern. **Communication networks** have shifted to data-only networks, meaning there are no more analog phone networks. This change implies that almost everything is vulnerable to potential attacks. It's not just wireless networks that can be targeted; even wired networks are susceptible to security threats. **Distributed services** are on the rise, requiring constant technical solutions to keep them running. This often involves outsourcing parts of server management, hosting, and adopting cloud services. This means that computers are no longer confined within a company, which necessitates trust in the service providers. Additionally, software development is getting more complicated due to various factors like software layering, framework integration, and the use of multiple programming languages. This complexity increases the chances of errors and vulnerabilities. In terms of security, the challenges can be summarized by the first engineering axiom:

"The more complex a system is, the harder it is to ensure its correctness."

Therefore, it's essential to keep systems as simple as possible. For instance, the number of bugs in a program tends to increase more than proportionally with the number of lines of code. The current complexity of information systems favors attackers, who can discover increasingly ingenious and unforeseen attack paths.

To express this idea clearly, we follow the **KISS rule**: "*Keep it Simple, Stupid.*"

1.3 A definition of ICT Security [OLD]

Each of us has a different concept of security: for example, the obligation to use safety belts when driving depends on country to country. Security is a personal concept but in engineering, we need to provide some formal definitions. Cybersecurity is a distributed part of a company, and every employee of a company must have the awareness for cybersecurity.

1. "Cybersecurity is the set of products, services, organization rules, and individual behaviors that protect the ICT system of a company."

Let us explain the keywords in the definition.

Products: refers to something that people can buy (such as products for firewall and VPN);

Services: these services are implemented by buying products;

Organization rules: they are required because even if, for example, a new system is set up with a password, rules must provide information to employees on how complex the password must be; otherwise, there will be no rules but personal behaviors, which could make the use of technical solutions less effective.

2. It is the duty to protect the resources from undesired access, guarantee the privacy of information, ensure the service operation, and availability in case of unpredictable events (**C.I.A. = Confidentiality, Integrity, Availability**).

More in detail, these three concepts have the following meaning:

- **Confidentiality** covers two related concepts:

Data confidentiality: assures that private or confidential information is not made available or disclosed to unauthorized individuals.

Privacy: assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

In terms of requirements and the definition of a loss of security, it means preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.

- **Integrity** covers two related concepts:

Data integrity: assures that information (both stored and in transmitted packets) and programs are changed only in a specified and authorized manner.

System integrity: assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

In terms of requirements and the definition of a loss of security, it means guarding against improper information modification or destruction, including ensuring information non-repudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.

- **Availability:** assures that systems work promptly, and service is not denied to authorized users. In terms of requirements and the definition of a loss of security, it means ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system.

This definition is a good starting point for cybersecurity, but more than the C.I.A. triad is required; two of the most mentioned are as follows:

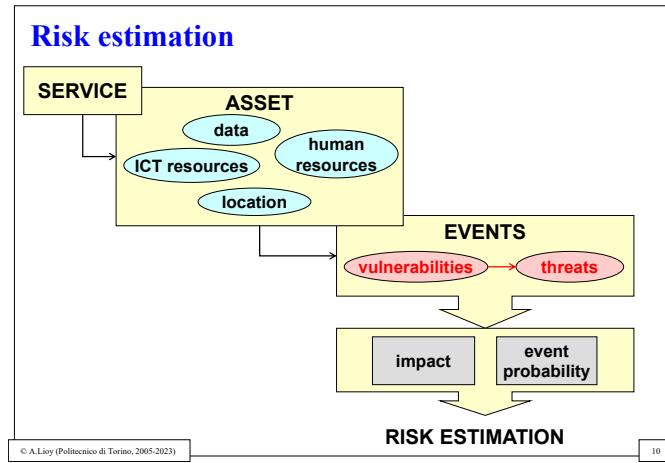
- **Authenticity:** the property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator. This means verifying that users are who they say they are and that each input arriving at the system came from a trusted source.
- **Accountability:** the security goal that generates the requirement for actions of an entity to be traced uniquely to that entity. This supports non-repudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action.

1. "The objective is to guard the information with the same professionalism and attention as for the jewels and deposit certificates stored in a bank caveau".
3. "The ICT system is the safe of our most valuable information; ICT security is the equivalent of the locks, combinations, and keys required to protect it".

1.4 Risk estimation

Before setting up a defense, we must understand what the risks are. To make a risk estimation, it is good to start from the **service**. Once we know the service we need to protect, we must identify the assets used to provide that service, and there are four categories of assets: **ICT resources** (computers, disks, networks), **data** (not the disks but something intangible that could be deleted or modified), **location** (assets must be inside a protected room), and **human resources** (which means the group of people who possess the knowledge that must not be shared).

After considering the assets, the next step is to identify the events that could affect their normal operation. The first point is that each asset has some **vulnerabilities** (for example, disks that are vulnerable to physical damage like a hammer hitting the disk), and some vulnerabilities can pose a real **threat** depending on the environment. For example, if a disk is left in an open place, someone might use a hammer to damage the disk. However, if the disk is locked in a room where nobody can access it, the vulnerability still exists but is not a real threat.



So, the process of analyzing a service searching for risks take place as follows:

- Find the **assets** of the service to be protected;
- Finding the **vulnerabilities** of each asset;
- Finding the **threats**, giving the way in which the assets are used;

Once we identify the threats we must:

- decide for each threat which **impact** it could have (what happens if disk is destroyed? If there's only one copy it could be a disaster, but if there are many that's not a problem)
- the **event probability** of the threat. This is the last point to get the **risk estimation**.

Recap of terminology:

Asset: the set of goods

Vulnerability: weakness of an asset;

Threat: deliberate action/accidental even that can procure the loss of a security property exploiting a vulnerability;

Attack: threat occurrence (deliberate action);

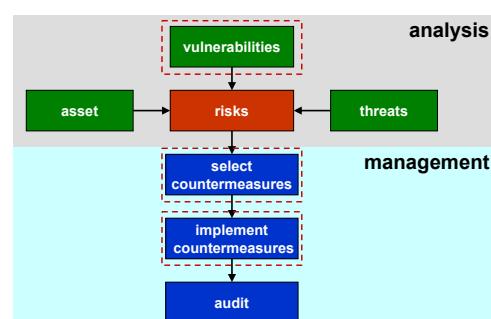
(negative) Event: threat occurrence

1.5 Risk management

To prioritize the risks, we can build a **risk assessment matrix** (or risk heat map):

catastrophic (5)	5	10	15	20	25
significant (4)	4	8	12	16	20
moderate (3)	3	6	9	12	15
low (2)	2	4	6	8	10
negligible (1)	1	2	3	4	5
	improbable (1)	remote (2)	occasional (3)	probable (4)	frequent (5)

1.6 Analysis and management of security



Risk estimation is only a piece for analysis and management of security. We can see that **assets**, **vulnerabilities**, **threats** give us the **risks** and then we start the **management** of the security. This means that for those risks that are not acceptable, either because they have high impact or high probability to end up in an attack, we need to select countermeasures and implement them. The last step is **audit**, that means that some independent person comes to check our work (if we correctly identified risks, selected the correct countermeasures, implemented them correctly).

In this course we will consider three of these blocks: vulnerabilities, the available countermeasures, and how to implement countermeasures.

What is the correct step in the lifecycle of a system to implement security?

In brief, there is no single correct point for implementing security because it must be addressed at each stage of the design process.

In more detail, when we perform the **analysis of requirements** for our system, we must conduct a risk assessment; based on these risks, we can define security policies and procedures that we will apply throughout the rest of the system design.

When evaluating **technical options**, we also need to identify security products. For example, when choosing a database, we must consider security alongside other factors such as speed and cost. If we opt for a database that automatically encrypts data, we have already addressed a security concern. Conversely, if we choose a faster database that lacks an encryption system, we will need to design that separately, incurring additional costs and efforts.

When **designing** the services that the system will offer, we must also include the security services component. Security should be integrated at each stage of the design process, not added as an afterthought. If we create a prototype website or app without any security measures, it will be challenging to retrofit security later on.



1.7 Relation in the security field

During the development of our system, we must integrate security at each step and ensure that the design is correctly implemented. We must test the system, including its security aspects. An example of this is testing against unexpected inputs to ensure that the system does not accept and process incorrect inputs.

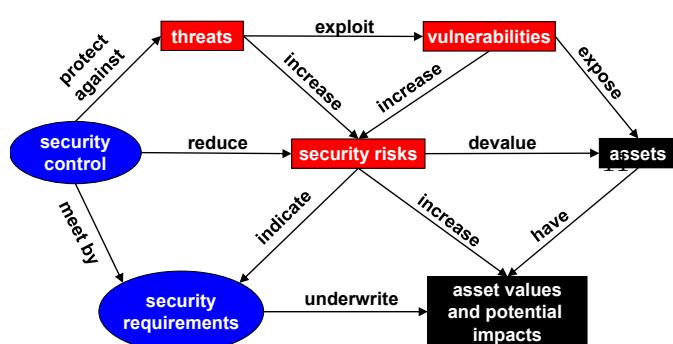
While implementing our system, it is important to establish security mechanisms. Some systems include security functions, but they are often deactivated due to associated costs (generally, activating security may slow down the system). For instance, the default password of a home router may be shared among different units, so it is necessary to change that password.

When the system is operational, security must be managed on a daily basis, as the security landscape changes constantly.

1.8 Relation in the security field

The black box is the system itself. Assets are exposed to vulnerabilities, and those vulnerabilities increase security risks. Additionally, threats exploit vulnerabilities, and the existence of vulnerabilities creates the opportunity for a threat.

On the left, there are the security requirements that we want to implement. Secu-



rity requirements are indicated by security risks, and security requirements are met by security controls.

The **security control** is the most important piece of the picture nowadays. It is an element placed in the system to protect against a specific threat and reduce the risks to which the system is exposed. Examples of security controls include firewalls, VPNs, and disk encryption.

Some terminology:

Incident: A security event that compromises the integrity, confidentiality, or availability of an information asset (generic definition).

(Data) breach: An incident that results in the disclosure or potential exposure of data.

- Disclosure: Occurs when data is intentionally given to someone.
- Exposure: Data becomes available to anyone who knows where to find it.

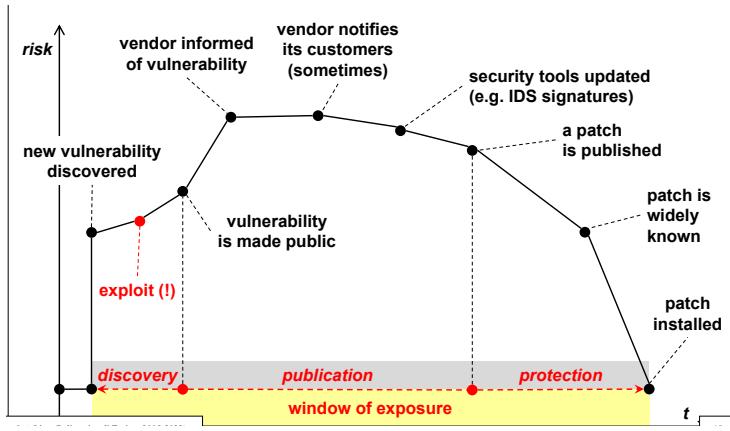
(Data) disclosure: A breach for which it is confirmed that data was actually disclosed (not just exposed) to an unauthorized party.

The difference between the last two is that the last one is a breach involving data that were not just exposed but also confirmed to be disclosed to an unauthorized party.

1.9 Window of exposure

The **window of exposure** is *the time between the discovery of a new vulnerability and the installation of a patch.*

Analyzing the graph from the left, we notice a consistently low level of risk, which, although it can never be reduced to 0, remains close to it. However, at a certain juncture, a new vulnerability is discovered, causing the risk to surge due to its uncontrollable nature. At a specific point (marked as the red point), an individual exploits this vulnerability to execute an attack. This action leads to the vulnerability becoming public, thereby making it accessible to everyone. This initial stage is commonly referred to as **discovery**.



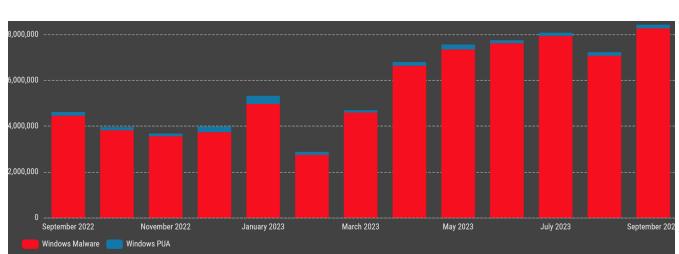
After an attack is carried out, two distinct groups of individuals become aware of it: the *bad guys* who aim to compromise the system and the *good guys* who strive to safeguard the system. Within these two categories, there is also the product vendor, who must be informed of the vulnerability. The vendor should, in turn, promptly notify its customers of the newly discovered vulnerability. While the vendor is working on fixing the vulnerability, users of that product should refrain from attempting to fix it, which is typically impossible. Instead, they should focus on updating their security tools, at the very least, to detect if the vulnerability is actively being exploited in attacks. This is the **publication** phase during which the vulnerability becomes public knowledge, and everyone is awaiting a fix while also making efforts to detect potential attacks.

Finally, at some point in time, the vendor creates a patch to fix the vulnerability. However, the patch must be distributed, and the risk only decreases when the patch becomes widely known and is eventually installed, rendering the system **protected**. The window of exposure can persist for days or even months, which is why security is an ongoing effort.

1.10 Some statistics

The graphic shows that there are about 10 million attacks per month using malware. That is why we should always keep our anti-virus/anti-malware updated. If we consider web servers across all sectors, 44% of servers were consistently vulnerable to attacks every day of the year.

The term **0 day** refers to the first day





when a vulnerability is reported to the public and has not yet been fixed.

The banking sector boasts the best security track record, with the lowest vulnerability rate (rarely vulnerable for 30 days or less each year).

In general, achieving strong security is challenging. Vulnerabilities can be discovered by both well-intentioned individuals (good guys) and malicious actors (bad guys). Some people investigate software to uncover vulnerabilities and inform the vendor to patch them before they are exploited by malicious actors. This proactive approach is taken to ensure patches are in place before vulnerabilities are discovered by the bad guys.

The "**0-day initiative**" (**ZDI**) discovers vulnerabilities and notifies the relevant organizations *before* making them public. ZDI typically provides a 120-day grace period from the discovery of a new vulnerability to allow the vendor to fix it before disclosing it to the public. Longer deadlines can be risky, as bad guys may discover the vulnerability during the extended time frame.

Example:

- 8 May 18: ZDI reports the vulnerability to the vendor, and the vendor acknowledges the report.
- 14 May 18: The vendor replies that they have successfully reproduced the issue reported by ZDI.
- 9 Sep 18: The vendor reports an issue with the fix and states that the fix might not be included in the September release.
- 10 Sep 18: ZDI issues a caution about a potential 0-day (which means that the vulnerability, for which a fix is not available, is going to be published).
- 11 Sep 18: The vendor confirms that the fix did not make it into the build.
- 12 Sep 18: ZDI confirms its intention to 0-day on 20 Sep 18.

1.11 Cyber threats

Components

There are **three main components** in cyber threats:

- threat actors (and their motivation)
- attack vectors (vulnerabilities and context)
- vulnerable targets (value for owner and attacker)



Motivations: *MICE*

What are the motivations behind this?

- **M is for Money:** direct transfer, blackmail, ... or indirect (e.g. data reselling);
- **I is for Ideology:** political, religious, hacktivism;
- **C is for Compromise:** individuals with no choice due to blackmail or threat against their families or themselves;
- **E is for Ego:** bragging around and positive reputation, "we do it because we can".



Figure 1.1: Threat actors

Standardization bodies

Standardization bodies in cybersecurity are organizations that develop and publish standards and guidelines to enhance security and interoperability in computer systems and networks.

(Cybersecurity) Standardization bodies (I)

- ISO (International Organization for Standardization)
- ITU-T (International Telecommunication Union, Telecommunication standardization sector)

- ISOC (Internet Society)
 - IETF (Internet Engineering Task Force)
 - IRTF (Internet Research Task Force)

- NIST (National Institute of Standards & Technology)
- ANSI (American National Standards Institute)

(Cybersecurity) Standardization bodies (II)

- ETSI (European Telecommunications Standards Institute)
- CEN (European Committee for Standardization)
- CENELEC (European Committee for Electrotechnical Standardization)

- BSI (British Standards Institution)
- UNI = Italian national body for standards (unification)
 - UNINFO = UNI body for information technologies and their applications

- generically named SDO (Standards Developing Organization), or SSO (Standards Setting Organization)

1.12 What is Security?

"Security is a process, not a product".

(Bruce Schneier, Crypto-Gram, May 2005)

If we have learned anything from the past couple of years, it is that **computer security flaws are inevitable**. Systems break, vulnerabilities are reported in the press, and still many people put their faith in the next product, or the next upgrade, or the next patch. "This time it's secure," they say. So far, it has not been. **Security is a process, not a product**. Products provide some protection, but the only way to effectively do business in an insecure world is to put processes in place that recognize the inherent insecurity in the products. **The trick is to reduce your risk of exposure regardless of the products or patches**.

Security Principles

To make it possible, it is necessary to follow some security principles:

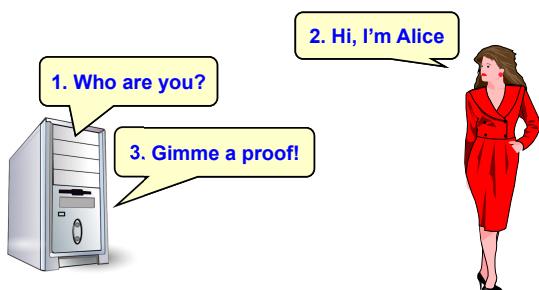
- **Security in depth:** if the enemy can defeat the first line of defense, there must be a second line to stop the attacker. Do not rely on just one defense, as that defense may have a bug or problem. It is better to have multiple levels of defense, so as the attacker breaks through the defenses, it will become increasingly difficult to keep penetrating.
- **Security by design:** this means that the security design is integrated into the system from the beginning and not added as an afterthought.
- **Security by default:** users should not have the choice to activate or deactivate security. Security should be enabled by default, and it should require significant effort to disable security features.
- **Least privilege:** this principle dictates that any element operating within the system should be assigned the minimum amount of privileges necessary to perform its task. Imagine a scenario where a system has excessive privileges, and it is being attacked by a virus. The virus could gain access to everything because of the excessive privileges.
- **Need-to-know:** this principle emphasizes that access to any component of the system should be granted only for the data required to execute a specific task. For example, in the case of Amazon, several people work within the system. When a customer places an order on Amazon, the first person handling the order can only see the details of what was ordered. They do not have access to information about who placed the order or the destination of the goods.

1.13 Security properties

Authentication (simple/mutual)	autenticazione (semplice/mutua)
Peer authentication	autenticazione (della controparte)
Data/origin authentication	autenticazione (dei dati)
Authorization, access control	autorizzazione, controllo accessi
Integrity	integrità
Confidentiality, privacy, secrecy	riservatezza, confidenzialità
Non-repudiation	non ripudio
Availability	disponibilità
Traceability, accountability	tracciabilità

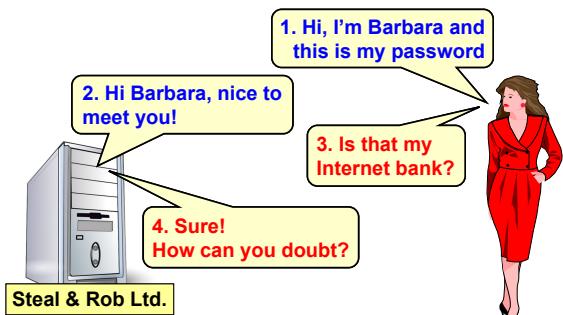
1.13.1 Peer authentication

Simple peer authentication



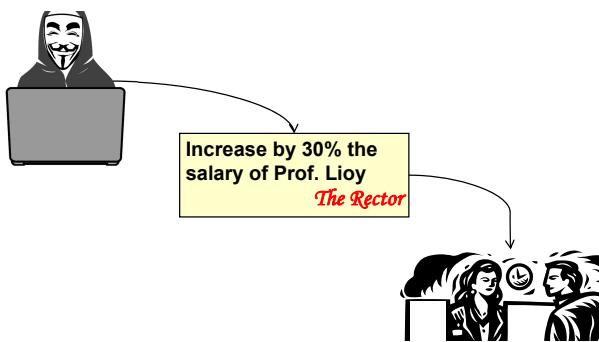
When there is communication between two peers that take part in the communication, they must authenticate. Two entities are considered peers if they implement the same protocol on different systems; for example, two TCP modules in two communicating systems. Computer systems usually ask some questions like the ones in the picture. Typically, for a proof, we provide a password. This is simple authentication: only one party authenticates. It attempts to provide confidence that an entity is not engaging in either masquerade or an unauthorized replay of a previous connection.

Mutual peer authentication



In mutual peer authentication, we also require a formal proof that we are connected to the real server. A fake server could display the same page as the ordinary bank just to obtain credentials from the user. When we connect to a server, we typically trust the server, but we need proof. Both parties authenticate each other.

Data origin authentication



Someone could write an email to request an increase in someone's salary, signed by the director, but there is no proof that the email came from the director. An electronic email typically lacks proper authentication. The same issue applies to files. Data are usually not authenticated, so there is a need for an authentication system for data as well.

1.13.2 Non-repudiation

Non-repudiation is a **formal proof** that is admissible in a court of law, providing **undeniable evidence of the data's creator**. Non-repudiation prevents either the sender or receiver from denying a transmitted message. This has several implications because we not only need **authentication** but also **integrity**. If someone alters data in a document, it should be detectable because it is no longer the original authenticated document.

There is a difference between authentication and identification. Authentication involves using electronic means to verify identity, such as a username and password. However, if the password is stolen, it raises doubts about whether it is the legitimate user or someone else. On the other hand, identification is much stronger, as seen in technologies like the Touch ID on smartphones, where the user is the only one capable of performing that operation.

Beware! The concept of "non-repudiation" is typically associated with not only technical aspects but also with specific procedures carried out voluntarily. We rarely achieve non-repudiation with protocols or procedures that automatically perform actions on behalf of the user.

Example of non-repudiation

Let's consider non-repudiation of an electronic signature:

- Syntax (is that your signature?)
- Semantics (did you understand what you were signing?) - what you don't understand has no legal value (for example the small lines in a document that are not understandable).
- Will (have you signed voluntarily?)
- Identification (was really YOU the signer?)
- Time (when did you sign?)
- Place (where did you sign?)

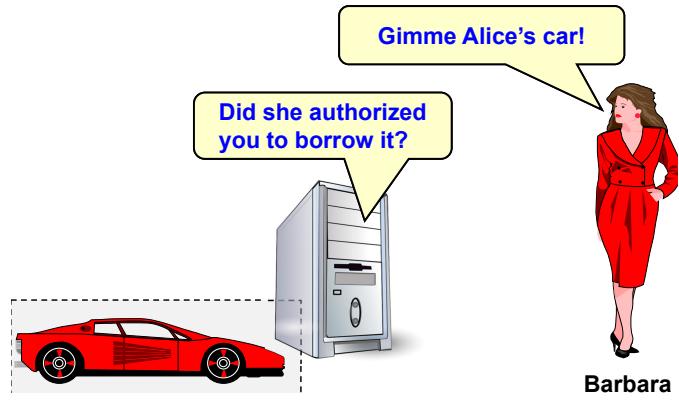
The electronic signature is a set of bits that represent the signature of a person. If we do not know where these bits must be placed to represent the signature, we cannot know which person the signature belongs to. For these reasons, there are specifications for electronic signatures.

1.13.3 Availability

An availability service is one that protects a system to ensure its availability. This service addresses the security concerns raised by denial-of-service attacks. It depends on proper management and control of system resources and thus depends on access control service and other security services.

1.13.4 Authorization (access control)

Authentication means identifying the actors in the system. After authentication, we can make decisions. For example, in the figure, Barbara has correctly identified herself and then asks the computer to "open the box of Alice's car" and the computer performs an **authorization decision (or access control)**. This is because the car in the box does not belong to Barbara. Therefore, a system must also determine if someone is authorized to perform an operation or not.



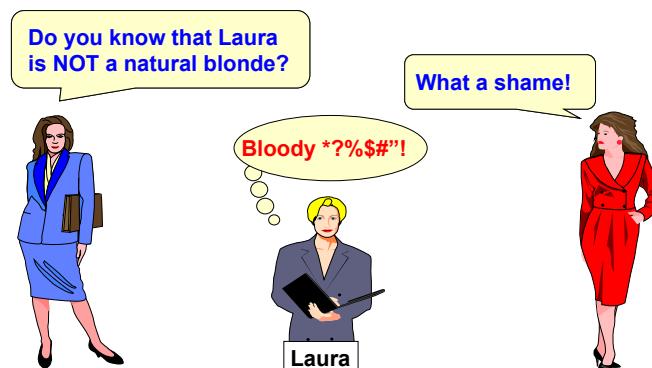
Important difference!

- **Authorization** is the process of verifying whether you, who have already been authenticated, are permitted to perform a particular operation.
- **Authentication** is the process of identifying the users of a system.

1.13.5 Privacy

Privacy (communication)

Privacy has several meanings. Communication privacy, for example, ensures that when two peers are communicating, it should not be possible for a third party to understand the communication. Even if someone can intercept the data passing through the network, no one (except the two end users) should be able to comprehend the contents of the communication.



Privacy (data, actions, position)

- **Data privacy:** Even if we have access to the physical location where data is stored, we might not be able to access the data.

- **Privacy of actions:** Companies have the right to monitor the websites visited by users, as do law enforcement agencies, especially in cases related to anti-terrorism laws. In Italy, all data sent over the network are stored for 7 years in case of future investigations.
- **Privacy of location:** Information about the location of users is available to those who manage the network. When defining the security properties of an application, we must consider concerns related to data, actions performed over the network, and potentially the location from which the communication originates.

1.13.6 Integrity

Integrity (data modification)



It means that if data have been modified, we are able to detect it. It doesn't mean that nobody can change the data; that's impossible. Network managers can always read and potentially modify data. That's why integrity refers to the detection of modified data.

Integrity (data cancellation/filtering)



Data can also be deleted, so we must ensure that if data deletion occurs, we are able to detect it. This is more challenging to detect because the receiver does not receive any notification (and does not have any hint that the payment shown in the picture should have been received).

Reply attack



Data sent over the network can be encrypted and thus made non-modifiable by the network manager. However, it is still possible to record the message sent over the network and replay it multiple times. Authentication will pass because the message is not modified, and it may not be easily understood by developers.

1.14 Data Protection

During the explanation of security properties, we always talked about protecting data. There are three types of data protection:

- **Data in transit:** when data are transmitted over a communication channel.
- **Data at rest:** when data are stored in a memory device.
- **Data in use:** when data are in RAM for use by a process.

Where is the enemy?

To defend something, we must know where the enemy is. There are a few possibilities:

- **Outside our organization:** In this case, perimeter defense using a firewall is required.
- **Outside our organization, except for our partners:** In this scenario, the firewall needs to be supplemented with protected paths or routes that allow communication between trusted users. This requires extranet protection, often in the form of a VPN (Virtual Private Network), which extends the intranet to include trusted partners.
- **Inside our organization:** In this case, we should focus on protecting the Local Area Network (LAN) and intranet applications, which can be challenging as we need to facilitate information sharing among users on the same network.
- **Everywhere:** Since attackers can be both inside and outside the organization, security measures must be implemented at the application level. Furthermore, since applications handle data, data protection must be independent of the physical location where the data is stored. For example, if a service like Dropbox is used but is not considered secure by the company, it is possible to encrypt the data before uploading it to Dropbox.

An example of this is an article from Corriere in 2009, which reported: "US PCs sold at the Peshawar market: Computers of the US army with restricted data sold for 650\$ along the road where NATO troops are attacked by the Taliban. Still full of classified information, such as names, sites, and weak points.

Threat model: where is the enemy? Which actions can it perform?

- **MITM (Main-In-The-Middle):** Sits between two peers A and B;
- **MATE (Main-At-The-End):** Resides inside one peer;
- **MITB (Main-In-The-Browser):** Resides inside one specific component of one peer (typically the web browser);
- **Passive Attacker:** Can only read the data/traffic;
- **Active Attacker:** Can read, but also modify, delete, or create data/traffic.

Basic problems (technological)

- The networks are insecure:
 - Most communications are made in clear (unless you take some actions);
 - LANs operate in broadcast (sending messages to everybody, and "if it's not for you, don't read");
 - Geographical connections are NOT made through end-to-end dedicated lines but through shared lines or through third-party routers.
- Weak user authentication (normally password-based);
- There is no server authentication;
- The software contains many bugs.

1.15 Some classes of attacks

Passive and active attacks

A useful means of classifying security attacks is in terms of *passive attacks* and *active attacks*.

A **passive attack** attempts to learn or make use of information from the system but does not affect system resources. Examples of passive attacks are:

- **Packet sniffing:** The content of network packets (e.g., passwords and/or sensitive data) is read by unauthorized parties.
- **Traffic analysis:** Even if the packet content cannot be understood by a third party, an opponent could extract some information about the nature of the communication taking place.

An **active attack** attempts to alter system resources or affect their operation, involving some modification of the data stream or the creation of a false stream. Examples of active attacks are:

- **IP spoofing / shadow server:** Someone uses the address of another host to take its place as a client (and hide its own actions) or as a server.
- **Connection hijacking / data spoofing:** Data is inserted/modified/cancelled during its transmission.
- **Denial-of-service / distributed DoS:** The functionality of a service is limited or disrupted (e.g., ping bombing).

1.15.1 Packet Sniffing (Eavesdropping)

Packet sniffing refers to reading packets addressed to another network node. It is easy to do in a broadcast network (e.g., LAN) or at switching nodes (e.g., router, switch). This is possible by putting the network card in promiscuous mode, which means that the card will read every packet passing through the device. This kind of attack allows intercepting anything (passwords, data, etc.). Some possible countermeasures are avoid using broadcast networks or encrypt the packet payload (if a non-broadcast network is not possible).



1.15.2 Traffic Analysis

Traffic analysis is more subtle than packet sniffing. Suppose we had a way of masking the contents of messages or other information traffic so that opponents, so that the content of the messages cannot be understood by a third party. Even with encryption in place, the opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

1.15.3 IP Spoofing (Masquerading)

IP spoofing means forging the source network address. Typically, the level 3 (IP) address is forged, but it is equally easy to forge the level 2 address (e.g., ETH, TR, etc.). A better name would be source address spoofing. This is typically used for attacks where an answer is not needed. If everybody is in the same subnet, due to the broadcast function, it is also possible to read the replies. Attacks of this type go for data forging and unauthorized access to systems. The countermeasure for this type of attack is to never rely on address-based authentication: generally, network addresses should not be trusted.



1.15.4 Denial-of-Service (DoS)

Denial-of-service refers to keep a host busy so that it cannot provide its services. For example, in public administration for a call to a competition, offers can be sent until a date. We can make an offer and stop all the others from sending email. A possible solution is to send tons of email keeping the server busy until the message "Message did not deliver because the destination mailbox is full". We saturated the mail service. Other examples:

- **Mail/log saturation:** As explained before.
- **Ping flooding ("ping bombing"):** The ICMP echo request usually uses a small number of bits (8 bytes) and starts a timer waiting some seconds for a response. We could use the largest amount of bytes possible: 64 Kbytes. We should send a lot of echo requests at maximum speed without starting timers. This will keep the host busy answering all these packets and prevent it from performing other tasks.
- **SYN flood:** It is a form of denial-of-service attack in which an attacker rapidly initiates a connection to a server without finalizing the connection. The server must spend resources waiting for half-opened connections, which can consume enough resources to make the system unresponsive to legitimate traffic (see TCP SYN flooding).

Usually, DoS attacks block the use of a system/device, and there are no countermeasures for it because it is not possible to know if someone connecting to the service is intentionally keeping the service busy or not. In other words, DoS is quite like a high increase in customers using that service. Monitoring and oversizing can mitigate the effects. Every time there is an alert that some threshold is passed, for example, resource saturation, it is time to investigate. It could be a system problem or a security problem. Security and system managers must work together.

1.15.5 Distributed Denial-of-Service (DDoS)

DDoS is essentially the same kind of attack as a DOS attack but magnified by the number of attackers simultaneously targeting a victim. Typically, a small number of individuals gain control of numerous nodes by installing DoS software on them. Each compromised node is often referred to as a **daemon**, **zombie**, or **malbot**, with the aim of creating a **botnet**.

A botnet is a network of compromised computers (*slaves*) controlled by a *master*. The master typically uses a **command and control (C&C)** infrastructure, which can be either client-server or peer-to-peer. The communication between the zombies (compromised computers) and the master is often encrypted or routed through "covert" channels. Covert channels are intended to mislead investigations; for example, information may be concealed by embedding it in **UDP packets** within **ICMP Request** messages, making it harder to detect since ICMP messages are common in normal network traffic.

These bots are sophisticated, and there have been cases where they can auto-update themselves to execute new attack techniques; by increasing the number of daemons (compromised computers), the impact of the attack can be magnified.

Other techniques to improve the attack include using a "**reflector**". This involves utilizing a potentially legitimate third-party component to relay the attack traffic to the victim. This approach offers two main advantages:

- **It hides the attackers' identities:** attackers send packets to the reflector servers with a source IP address set to their victim's IP (IP spoofing), indirectly overwhelming the victim with response packets.
- It **multiplies the effect** through an "**amplification factor**" N:1 (typically protocol-dependent), where the reflector server's response size is much larger than the request size. For example, when making a specific query to DNS, the DNS response can be up to 70 times larger than the request.

DDOS attack



There is an attacker and a victim. The attacker somehow creates a network of daemons, then selects some nodes to be masters (usually more than one for redundancy), which are part of the botnet. The attacker sends the IP address to attack and then *disconnects from the network* to avoid being tracked. The other masters will coordinate the work of the daemons, which must perform the attack at the same time. Masters do not directly take part in the attack; they try to remain hidden as much as possible to avoid being detected during an investigation because this would stop the attack. The huge class of amplification factors that daemons can create can overwhelm the victim.

Masters and daemons are not real individuals but part of the botnet. The only human involved is the attacker, who disconnects after initiating the DDOS attack.

Case history: DDoS towards Yahoo Server Farm

The first well-known attack of this kind occurred on Feb 8th, 2000, at 10:30 am (PST) against the Yahoo Server Farm. Administrators wrote a report that stated:

- "The initial flood of packets, which we later realized was in excess of 1 Gbit/sec, took down one of our routers..."
- "... after the router recovered, we lost all routing to our upstream ISP..." This happened because an internet connection requires two routers. Yahoo's people rebooted their own router, but at the other end of the communication link, the router was still down.
- "... it was somewhat difficult to tell what was going on, but at the very least, we noticed lots of ICMP traffic..."
- "... at 1:30 pm, we got basic routing back up and then realized that we were under a DDoS attack."

Later, the attack was traced back to the 15-year-old Canadian boy Michael Calce (aka MafiaBoy). Even if getting to the attacker is not always possible, a U.S. lawyer can easily identify who the daemons are and seek refunds from them, even if the daemon was a victim of the attacker, for example, if the attacker took control of a computer for DDoS purposes.

Case history: DDoS towards "Krebs on Security" Blog

On September 27, 2016, the administrator of the blog received a DDoS attack that generated 665 Gbps, and it was generated by a botnet of IoT devices (or claimed to be such). There was no use of any reflectors or amplifiers, just millions of devices that performed perfectly valid requests. It seemed like millions of users wanted to connect to the website at the same time. This generated a significant amount of traffic, and even though the blog was hosted on Akamai, the traffic was so substantial that the company had to give up and made the blog unreachable by dropping that destination from their routing table on September 29. There was no known reason for the attack; perhaps it is connected to Krebs' analysis of similar attacks against online game servers.

1.15.6 Shadow Server / Fake Server

In **shadow server** attacks, a host manages to impersonate itself as a service provider to victims without having the right to do so. There are two techniques to achieve this:

- If the attacker **can sniff requests and spoof responses faster than the real server**, the latter will not be able to communicate because the second package will be discarded, considering it a duplicate.
- **Routing or DNS manipulation**, mapping the real name to the IP of the shadow server.

The attacks that can be carried out in these scenarios include:

- **Issuing incorrect answers**, providing a "wrong" service to victims instead of the real one.
- **Capturing victim's data** provided to the wrong service.

The countermeasure to this type of attack is to **require server authentication**.

1.15.7 Connection Hijacking / Man In The Middle (MITM)

It is also referred to as **data spoofing**, and this form of attack involves gaining control of a communication channel to insert, delete, or manipulate the traffic. This can be achieved through logical means (by altering the network's routing) or physically (if one can physically access a router or switch).

These attacks can be executed for various reasons, such as eavesdropping, inserting false data, and altering data exchanged between two parties.

Countermeasures include **authentication**, ensuring data **integrity** (verifying whether it has been altered), and packet **serialization** (ensuring that no packets are added or deleted, and that packets are received in the same manner as they were sent) for each individual network packet.

1.15.8 Trojan

A Trojan is a program that hides a malicious payload within a seemingly harmless exterior. Despite the increasing security of network channels, user terminals have become more susceptible to such attacks. This vulnerability extends to devices like smartphones, smart TVs, and various Internet of Things (IoT) devices.

These attacks tend to target less tech-savvy or "ignorant" users. Attack tools can take on various forms, from traditional methods like embedding keyloggers within seemingly innocuous applications to more contemporary approaches, such as malicious browser extensions.

Trojans are frequently used to carry out two distinctive types of attacks: "**Man-At-The-End**" (*MATE*) and "**Man-In-The-Browser**" (*MITB*).

1.15.9 Zeus

The Zeus attack, also known as **Zbot**, is a major malware threat combined with a botnet. It was first discovered in 2007, and law enforcement has been actively seeking the owner of this network. The owner claimed to have sold it in 2010.

Zeus can be used in the following ways:

- Directly: For example, it can be employed for Man-In-The-Browser (MITB) attacks to perform keylogging or form grabbing (software that reads data in a form).
- Indirectly: It can also be used to deliver other malware, such as the CryptoLocker ransomware.

Zeus is challenging to detect and remove because it employs stealth techniques to conceal itself. In the USA alone, there are approximately 3.6 million active copies of Zeus.

1.15.10 Software Bug

Even the best software can have bugs, which can be exploited for various purposes. One common way to exploit a software bug is to create a Denial of Service (DoS) attack.

For instance, there was an attack against the WinNT server (versions 3.51 and 4.0) where attackers discovered that the server hosted a service on an undocumented TCP port 135. They attempted to communicate with this port by sending 10 random characters followed by a carriage return (CR). This caused the server to become unavailable, as it experienced 100% CPU load even though it was not performing any useful work.

The issue stemmed from the fact that at port 135, there was a new Microsoft-only service that was a remote procedure call (RPC) and had a bug. When it received a malformed packet, it went into an infinite loop, continuously asking, "where is a good packet?" Since the ARP server (?) was part of the operating system's kernel, it consumed 100% of the CPU without the possibility of interruption. Microsoft developed a solution with Service Pack 3 (SP3), in which they addressed and corrected this bug.

1.16 Virus & Co. (malware)

- **Virus:** A virus is a malicious program that damages the target and then duplicates itself. It is propagated by humans involuntarily.
- **Worm:** A worm damages the target indirectly by replicating itself to the extent that it achieves resource saturation. Additionally, a worm will attempt to propagate automatically. Worms are usually more challenging to detect because the damage they cause may resemble normal activity unless a careful analysis of the network pattern is conducted.
- **Trojan Horse:** The Trojan horse is a program that carries some malware. It may appear to be a valid program but can also install malware.
- **Backdoor:** A backdoor is a piece of software that provides an unauthorized access point. While illegal, it is common among developers. For example, if a developer is concerned about not being paid, they might create a backdoor to gain access in case of payment problems.
- **Rootkit:** A rootkit is a set of tools that provides privileged access. It is hidden, often disguised as a modified program, library, driver, kernel module, or hypervisor. Due to its stealthy nature, rootkits are challenging to detect and remove.
- **PUA (Potentially Unwanted Applications):** It's a sort of grayware, not directly dangerous.

Virus and worm (malware)

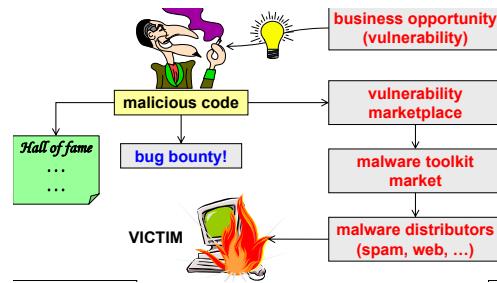
Virus and worms require some kind of complicity (may be involuntary) from **the user** (gratis, free, urgent, important), the **system manager** (wrong configuration), **the producer** (automatic execution, trusted).

Some countermeasures are user awareness, correct configuration/secure software, antivirus (installed and updated).

Malware food chain

When someone discovers a vulnerability, it can be exploited for the development of malicious code (for example, this code could be used to gain access to a website and manipulate its data). Such information can be exploited in two ways:

- Especially in the past some individuals may perform the attack primarily for the sake of recognition, like being in a "Hall of Fame".
- More recently, individuals have been motivated by business interests. They sell the information about the vulnerability or a working code that exploits it on the *vulnerability marketplace*, a hidden marketplace and is found exclusively on the internet, on dummy servers that appear for a brief time during the night.



Payment for these vulnerabilities is often made using cryptocurrency. The buyers of these vulnerabilities are typically the **malware toolkit makers**, which are used to develop attack programs that incorporate malware; malware distributors, such as spammers and website owners, then employ this type of software for their illicit activities.

Zeus: Cyber Theft Ring

It often happens to receive an email that asks you to send money outside of the country in exchange for a large sum of money. If you accept this request, you will supposedly receive the money. They may begin by testing you, sending a small amount of money, for example, \$10,000, and then ask you to send \$7,000 to Nigeria, allowing you to keep \$3,000 for yourself. However, by agreeing to this, you will be facilitating money laundering, as the funds are stolen from someone's bank account and transferred to you, making you appear responsible. You will then send the money via MoneyGram, while the individuals in Nigeria will disappear, leaving you to face potential legal consequences. This scenario is a typical case of acting as a 'money mule,' which involves accepting funds from a stolen bank account and forwarding them to the intended destination."

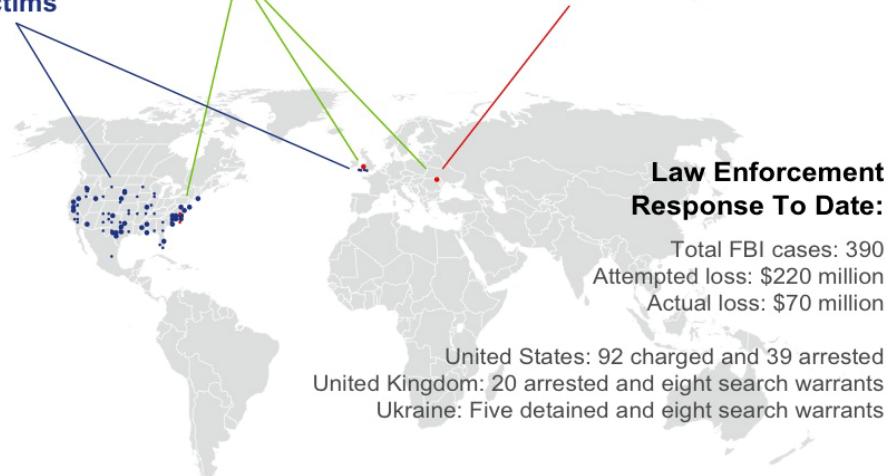
Cyber Theft Ring



source: http://en.wikipedia.org/wiki/File:FBI_Fraud_Scheme_Zeus_Trojan.jpg



source: http://en.wikipedia.org/wiki/File:FBI_Fraud_Scheme_Zeus_Trojan.jpg



source: http://en.wikipedia.org/wiki/File:FBI_Fraud_Scheme_Zeus_Trojan.jpg

Most of the victims are based in US or UK. The mule organization is spread across many countries: US, Europe, British Ireland, but malware coders/exploiters are typically in the East Europe. In the picture there are also numbers about the Law Enforcement Response.

Ransomware

Ransomware is a type of malware that is designed to extort a **ransom**, typically by demanding money in exchange for releasing something. When ransomware infects a computer, it usually encrypts the contents of the disk, rendering them unreadable. This also applies to tablets and smartphones, for example, by changing access passwords to data.

However, paying the requested ransom (usually in bitcoin) does not always guarantee the recovery of your data. This is because the key required to unlock the files is typically stored on a server that could be shut down as a result of a police investigation. As a result, the attacker may no longer have access to the key used for locking the data.

Ransomware-as-a-service

For some time, there was a *ransomware-as-a-service*, in which individuals provided the source code for ransomware; this enabled people to use ransomware even if they had little understanding of how it worked. This service was known as the **TOX malware** and operated on the **TOR anonymous network**. TOX would demand a ransom and handle the payment, charging a 20% service fee. The "customer" has the only task to distribute it to the victims (for example by leaving a USB pen in public places). TOX saw rapid growth, with around 1,000 customers per week and infecting approximately 100 devices per hour.

Even though TOX was eventually stopped by law enforcement, ransomware threats continue to exist. Technology alone is insufficient for protecting against ransomware. While having a robust anti-malware system installed is helpful, it's important to note that this type of attack targets not only end-users but also servers.

Ransomware: Not only technology but also procedures and organization

Ransomware typically **encrypts data** that can only be restored with the proper decryption key. Regular backups can provide a potential solution, but is that enough?

- **How old is the backup?** Consider the age of your backups. Even if you create daily backups, ransomware developers are aware of this practice and have developed a stealthy variant known as silent ransomware. *Silent ransomware* infiltrates your system but instead of encrypting your current data, it targets your backup files. This can continue for several days, rendering your backups unreadable, and it may only be discovered later that the backups are corrupted.
- **Offline or Network Backups?** Backups should be made offline. For instance, consider a case where a dentist's office used a Network-Attached Storage (NAS) system for continuous backup of local data; ransomware found this setup and encrypted the data on the network storage, rendering the backups useless.

To enhance security, an inverted backup technique is recommended. In this approach, the backup disk is not directly connected to the target PC (the one whose data is being backed up). Instead, there should be another node that remotely mounts the target PC's disk, makes a copy, and then disconnects from the target PC. Even if the target PC is infected, this process remains unaffected. Importantly, the backup node should not be connected to the network to prevent potential malware interference.

- **Verified or "Trusted" Backups?** Have you verified the correctness of your backups, or have you merely trusted them? There is a well-known case in Sweden where magnetic tapes used for backups were corrupted during transportation due to heated seats generating currents on wires, creating magnetic fields that damaged the tapes. This issue was only discovered when data corruption occurred, and there was a need to access the backups.
- **When was the attack?** To identify the correct backup, it's essential to know the timestamp of the attack. There have been cases, such as a video archive, where determining the accurate backup was challenging. In this example, a journalist looking for video clips noticed inconsistencies in file names. While backups existed, determining which one was correct depended on when the attack took place. Without knowledge of the attack's timing, selecting the right backup for restoration becomes impossible.

1.17 Basic problems (non-technological)



Figure 1.2: Even the most sophisticated technological tools can be made useless by human error

- **Low Problem Understanding (Awareness):** People often lack awareness of security issues and tend to implement security measures only after experiencing problems. This behavior stems from the belief that such issues will not affect them.
- **Human Mistakes (Especially When Overloaded or Stressed):** Errors are more common when individuals are under stress or overwhelmed with work.

- **Inherent Human Trust:** Humans tend to have a natural inclination to trust others, especially in online interactions where face-to-face contact is absent.
- **Complex Interfaces/Architectures Can Mislead Users and Lead to Errors.**
- **Performance Decrease Due to the Application of Security Measures:** Consider the example of a company that sought Lioy's assistance with antivirus software. Although the company claimed to have the best antivirus, they frequently experienced virus infections that disrupted their system. Lioy visited the company's premises and examined everything, including antivirus installation and updates. While exploring the office to understand the problem, he noticed a desktop with the antivirus icon disabled. Lioy inquired about this, and an employee explained that the antivirus's file scanning process caused delays in accessing files, prompting employees to disable it temporarily for more efficient work. Surprisingly, all employees followed the same practice. Lioy concluded that the company had a non-technological problem.

Social engineering techniques and strategies

In the context of information security, **social engineering** is the psychological manipulation of people to induce them into performing actions or divulging confidential information. Typically, the targets are **naive users**, such as those who may be urged to "immediately change their password due to an alleged PC attack". However, even experienced users can be targeted, for instance, by replicating a genuine email but altering its attachment or URL.

These tactics are employed through various channels, including email, phone, fax, and even physical documents; they often rely on **psychological pressure**. Usually, this is achieved in two ways:

- By exploiting human empathy towards a friend or a co-worker, for example generating a fake message conveying a message like "Help me, otherwise I'll be in trouble...", leading to some action that could be against best practices, creating an opportunity for an attack.
- By pretending to be a boss, also asking for something that could be forbidden, under the threat "do it, or I'll report it to your boss...".

In general, those who run these psychological attacks try to show familiarity with the company's procedures, habits, and personnel to gain trust and make the target lower his **defenses**.

Phishing(pronounced "fishing")

It is an attack technique based on attracting a network service user to a fake server (shadow server), using mail or IM, for acquiring his authentication credentials or other personal information, or persuading him to install a plugin or extension which turns out to be a virus or a Trojan. An example message can look like: "Dear Internet banking user, please fill in the attached module and return it to us ASAP according to the privacy law 675...". There are some variants:



- **Spear phishing**, when the message includes several personal data to disguise the fake message as a legitimate one (e.g., email address, name of Dept/Office, phone number, etc.)
- **Whaling**, when the target user holds a position of responsibility or power, such as a CEO or CIO. This has two main advantages from the attacker's point of view: the damage done by deceiving these types of people can be much greater, as their credentials can give the attacker great power; these people generally occupy a commercial and administrative position, which usually means that they do not have the necessary knowledge to handle security issues.

Pharming

It's a term of controversial use and embodies a set of several techniques to redirect a user towards a shadow server by:

- changing the "hosts" file at the client.
- changing the nameserver pointers at the client.
- changing the nameservers at a DHCP server (e.g., an ADSL/Wireless router).
- poisoning the cache of a nameserver.

This could happen via a **direct attack** if there is a vulnerability or misconfiguration, or via an **indirect attack** with a virus or worm.

Fake Mail / IM

Creating a fake email is relatively easy, but maintaining the right "tone" can be challenging. For instance, questions like "Should I sign the emails as Antonio Lioy, or just Antonio, or A. L.?" need to be addressed, as an incorrect tone can lead to the detection of the fake email. The creation of fake SMS or instant messages (IM) is also possible, allowing for various deceptive scenarios:



- *Fake ATM withdrawal* messages that prompt the recipient to call a specified number, which then solicits their confidential credentials.
- *Fake kidnapping alarms*, often delivered via IM. It is not uncommon for individuals, particularly older men, to approach the police claiming, "Help me because my daughter has been kidnapped, and they demand 200,000 euros for her release." In reality, the person making the request is often a woman met on social media who has established a fake relationship with the victim to elicit empathy and financial support.

Case history: "Mr. Confindustria in Brussels tricked by a hacker: 500,000 Euros lost. Fired." (Repubblica journal on 30th September 2017)

"Transfer immediately half a million to this foreign bank account." However, this email was sent by a hacker, and the money disappeared. The fake order was (seemingly) signed by Director Panucci: "Please execute and don't call me because I'm out of the office with the president."

Final message: all the employees should be trained in the security problems of modern life.

Case history: T.J. Maxx attack (2007)

In 2007, there was an attack against T.J. Maxx, a chain of shops. The attack occurred externally, allowing the attacker to access 45 million credit card numbers from customers. The attack lasted for about 18 months, ending in January 2007. It gained notoriety when a group of 300 banks, affected by the attack as they had to refund customers, initiated a class action lawsuit against T.J. Maxx, seeking a \$10 million refund. This was due to the fact that, even though it was known in 2007 that the wireless protocol WEP was insecure, T.J. Maxx continued to use WEP instead of WPA. The attacker didn't need to physically enter the shop; instead, they could sit in a nearby park with a laptop and appropriate software, intercepting all transactions between the cash register and the back-end server. The attack was carried out by 10 people (3 USA, 3 UKR, 2 CHN, 1 BEL, 1 EST + "Delpiero") with the assistance of an ex-hacker who had been hired by the US Secret Service.

This practice is not uncommon; when an attacker is apprehended, they are often given the opportunity to work for the government as a consultant or to help prevent future attacks."

Considering the legislation, in general, the law does not prescribe specific security measures to be taken, but merely advises establishing defenses in accordance with the state of the art. This means that a security manager must always stay up-to-date and apply the most recent solutions against new types of attacks.

Case history: US Air Force phishing test transforms into a problem (04/2010)

In April 2010, during an Operational Readiness Exercise (ORE) at Andersen Air Force Base located on Guam Island, a phishing email was used to test the response of airmen. The email, sent by security testers, claimed that film crews were starting production of 'Transformers 3' on Guam and invited airmen to submit applications on a website if they wished to work on the shoot. The website then requested sensitive information, which the airmen should have been trained not to provide. The outcome of this exercise became widely known because one of the airmen posted the news outside the base, reaching the civilian world. As rumors spread that the highly anticipated film was coming to Guam, local media began contacting the base, leading to efforts to clarify the situation.

Andersen AFB's leadership expressed regret for any confusion caused by this phishing exercise. They emphasized the importance of individuals being cautious about the genuine threat posed by phishing emails, with the hope that others can learn from this exercise.

1.18 Some important recent attacks

The first attack discussed is **Stuxnet**, known for being the first cyber-attack to cause physical damage. Next is **Black Energy**, one of the most well-known attacks used against critical infrastructure, such as the electrical grid distribution. Finally, we have **Mirai**, **BlueBorne**, and **BrickerBot**. These attacks are all targeted at IoT devices, embedded systems, the automotive sector, and home devices.

1.18.1 Against Cyber-Physical Systems

Stuxnet (2010)

It is important because it is the prototype of a new kind of attack, since it caused physical damage (like a virus) and it attempts to propagate itself to other systems (like a worm): the target were **SCADA systems** of a specific manufacturer attached to the infected nodes. SCADA stands for Supervisory Control and Data Acquisition, and they are those computer systems that control process plants or machinery, for example production systems for cars, food, and so on. Since SCADA is the interface between computer and physical reality, an attack against it could damage physical system components. Stuxnet was a very sophisticated attack because this worm contained four attack vectors:

- one was based on an already known vulnerability (for which a patch existed);
- one exploited an already known vulnerability, but for which no patch was available (for those who got the patch for the first one);
- two "zero-day" vulnerabilities (if there were countermeasures for the known vulnerability).

Attack mode It seems that the first infection came through a USB pen inserted in one of the computers that managed the SCADA system. Then it propagated through all the computers connected to the network of the enrichment plant in Iran thanks to shared disks, Microsoft Spooler Subsystem and RPC services bugs. The infection via USB key was possible because inside that network of computer there was no internet connection, and a USB Key was required for updates of SCADA systems. It seems that the technician arrived from another country for the maintenance with the USB pen (and good software in it), got distracted in the hotel from some girls while someone added to the USB pen the malware. This malware was disguised as a driver (pretended to be a driver) with a digital signature validated by Microsoft and used two different certificates. Additionally, the guilty was not only from Microsoft but also of the developers, because the SCADA system had only one shared default password for the back-end database.

Attack motivations Even the timing and the location are peculiar. First, on 17/6/10, the first damage was created, and it became known that Stuxnet existed. For this reason, an investigation started, and one week later, on 24/6/10, the analysts detected the use of the first signature certificate and revoked it on 17/7/10. However, the malware then started using a second signature certificate. Since its discovery, it took one month for security bulletins by CERT (Computer Emergency Response Team) and MS (Microsoft). Then, Microsoft started developing patches that were gradually released through October '10. This malware stopped propagating itself on 24/6/2012 (which was written inside its code). This suggests that the malware was targeted to a specific time window.

Most of the attacks are against the most developed countries, but this malware was found in 52% of cases in Iran. It seems to be targeted at a specific area: *this worm had the target of destroying the uranium enrichment plant in Iran.*

Attack consequences The malware was able to manipulate the speed of the cylinders used for uranium enrichment beyond safe limits, all without triggering any alerts in the monitoring system used by the technicians. As a result, the technicians only managed to halt the system after significant damage had already occurred. This setback delayed the Iran Uranium Program by several years in their efforts to restore the proper functioning of the facilities.

Lessons learnt from this attack:

- Physical system separation (air gap) does not imply security: in fact, without any other standard protections (anti-virus, OS patches, firewall), these systems are not secure. In this specific case, the systems did not use any kind of standard protection, relying too much on the physical barrier;
- Unnecessary active services can be a source of vulnerabilities: MS-RPC, shared network print queues, shared network disks were not necessary but still running. In general, any unnecessary service should be disabled;

- A validation list for software to be installed should be used: it seems that the technician copied all the content of his USB pen, also copying the added malware. He should have copied only the file necessary for the SCADA systems update.

1.18.2 Against critical infrastructure

Fancy Bear / APT 28

This hacking group is believed to have ties to the Russian military intelligence agency, GRU. Over the years, they have been involved in cyber-attacks and espionage activities against various high-profile targets; some of their notable targets include the German and Norwegian parliaments, the French television network TV5Monde, the White House, NATO, the U.S. Democratic National Committee, and the Organization for Security and Co-operation in Europe (OSCE).

Additionally, during the years 2014-2016, they developed Android malware specifically designed to target the Ukrainian Army's Rocket Forces and Artillery.

Against Internet-of-Things, automotive, home

Mirai

Mirai is the most famous IOT **cyberworm** discovered around September-November 2016. This kind of attack does not damage directly but simply saturates the resources (like the DoS). Every system infected by Mirai becomes part of a huge botnet for a large-scale DDoS. It was used for disruptive attacks such as:

- Krebs on Security → up to 620 Gbit/s
- Ars Technica → up to 1Tbit/s
- Dyn DNS provider → requests from tens of millions of IPs

Botnets are deployed into millions of IoT devices such as cameras, residential routers, or baby monitors because these devices have almost no installed protection. In this way, Mirai can easily spread into home networks. The fast networks (such as 4G and 5G) make it even more critical because these devices can send a lot of data per second. Mirai is a very complex malware because it has almost no external dependencies (this means that the malware is compiled with static libraries) and it is cross-compiled to be executed on several platforms. Mirai is released as an open-source worm: this means that not only it is possible to study it, but also that new variants can be easily developed by other attackers. Mirai contains a propagation scan phase to compromise additional targets (starts contacting other nodes if it can infect them). It also observes the victim system before contacting the C&C: when Mirai is installed on a device, it doesn't know if it is a true or fake device (maybe used to study Mirai's behavior) and for this reason, when it starts, it connects to a fake C&C. IF the communication works, it means that there is a problem (because the connection should not work, which means that someone is giving Mirai full access), and it shuts down automatically. Only after performing this check, it THEN contacts the real C&C.

The kind of attacks, since it is a DoS network-based, is driven by C&C that can decide which attack to make and can open many TCP connections, send tons of UDP packets, or use GRE or simple SYN flooding.

BrickerBot

In October 2017, a message appeared on blogs and chat platforms (as shown on the left side). In this particular case, Wind, a telecommunications company, was faced with an issue where they had to physically replace all the modems. They were unable to resolve the problem remotely. The modems had malfunctioned and needed replacement. The individual responsible for this attack claimed to have done it for the greater good, possibly to prevent the formation of a botnet. This incident bore some similarities to the Deutsche Telekom case in 2016, albeit with more severe consequences.

A screenshot of a messaging application interface. On the left, there is a small profile picture of a person. To its right, the name "Bonny F." is displayed in blue, followed by the text "un'ora fa". On the far right of the message area, there are two small icons: a square with a minus sign and a square with a vertical line. The message content itself is enclosed in a red border and reads: "Hi guys.. Sorry for not speaking Italian. The Wind modems had telnetd running on port 8023 and a default password admin/admin which gave anyone root access to them. Unfortunately the modems got bricked by malware known as 'BrickerBot' which wrote random data over the partitions. When Wind eventually asks customers to return the devices for a replacement you'll want to be first in line.."

Lessons learnt:

- **Use strong passwords;**
- **change default password** upon installation;
- **Permit external administrative access only from specific “trusted” networks** (and not from any address on internet): using IP addresses as “filter” to allow access is anyway weak, but at least is better than allowing it from all;
- **Apply timely all security patches to minimise the WOE:** in the Wind-Infostrada case, the bug was well-known, so the company should have installed the patches.

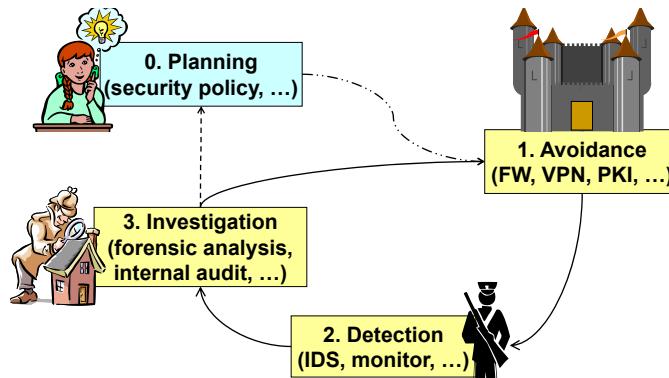
1.19 Attack maps: sample data sources

There are many websites that can show how many attacks are being performed. It is possible because many of these sites are providers of security solutions. They get the information from their own software which is installed on millions of devices and detects possibly attacks. The websites have many labels which tell exactly what attack is:

- **OAS = On-Access Scan:** Anti-malware that performs a scan to detect malware before accessing a file.
- **ODS = On-Demand Scan:** Scans inserted USB pens before files are copied.
- **MAV = Mail Anti-Virus:** Antivirus for emails that checks attachments.

- **WAV** = *Web Anti-Virus*: Before downloading something from the internet, the item is temporarily stored in a central facility, checked for possible malware, and only transmitted to the destination if nothing is found.
- **IDS** = *Intrusion Detection System*: Monitors networks and hosts for possible attacks.
- **VUL** = *Vulnerability Scan*: More pro-active; some machines periodically try to contact others on the network and check for vulnerabilities. If something is found, a warning is sent.
- **KAS** = *Kaspersky Anti-Spam*: Specifically for Kaspersky.
- **BAD** = *Botnet Activity Detection*: Detects botnets performing activity.

1.20 The three (four) pillars of security



1. **Planning** (security, policy, ...);
2. **Avoidance** (FW, VPN, PKI, ...);
3. **Detection** (IDS, monitor, ...);
4. **Investigation** (forensic analysis, internal audit, ...);

1.21 The NIST cybersecurity framework



Figure 1.3: <https://www.nist.gov/cyberframework/online-learning/five-functions>

1.22 Final slides



Kevin Siers, NC, USA (cartoon from the Charlotte Observer)

Chapter 2

Cryptographic techniques for cybersecurity

2.1 Cryptography



The most used technique to achieve protection for many centuries is **cryptography**; a mathematical technique that involves algorithms for encryption and decryption:

- the encryption algorithm takes a message (in clear) and transforms it in such a way that it becomes unintelligible;
- to recover the original text, the decryption algorithms make it readable again.

Next to the algorithms, **key-1** is needed for encryption and **key-2** for decryption, both of which are streams of bits. Cryptography is used in communication and for data storage (for example, to store data on disks without permission to read them except for authorized users). The common terminology used in cryptography includes two other keywords:

- **Plaintext** or **cleartext**: the unencrypted message, typically referred to as **P**;
- **Ciphertext**: the encrypted message, typically referred to as **C**. Note that in some countries, the term "encrypted" may sound offensive for religious reasons (related to the cult of the dead); in such cases, "*enciphered*" is preferred.

Cryptography's strength (Kerchoffs' principle)

Kerckhoffs' Principle (1883) states that the security of a cryptosystem must lie in the choice of its keys only; everything else (including the algorithm itself) should be considered of public knowledge. However, this principle relies on the fact that the keys have the following properties.

- Are kept **secret**;
- Are managed only by **trusted systems**;
- Are of **adequate length**

If these properties are met, not only it has no importance that the encryption and decryption algorithms are kept secret, but it is better to make the algorithms public so that they can be widely analysed, and their possible flaws and vulnerabilities identified.

Security through obscurity (STO)

The Kerckhoffs' Principle is related to the concept of **Security through obscurity**: it means that a system is protected, but the details on how it has been protected are not disclosed. Generally, this alone is not considered a valid security mechanism because if someone discovers how the system has been protected (and we have seen that there are also non-technical ways by which this can be achieved), it is no longer secure.

For this reason, we say that "*Security through obscurity is as bad with computer systems as it is with women*".

"Men try to hide things from women, but when they discover the truth, it is worse than if they had known it from the beginning"

(Antonio Lioy)

Editor's note: don't hide things from your partner, regardless of gender.

However, there is a category of people (such as military men) which tend to apply STO, but as an additional layer. It is possible to use STO as a layer only if a really strong algorithm is used (but not a secret one).

2.2 Symmetric and asymmetric cryptography

Depending on which relation exists between key-1 and key-2 there are different kinds of cryptography.

Secret key / symmetric cryptography

It is so named because **only a single key** is shared by the sender and receiver. In the diagram, there is a plaintext used as input for the encryption (E) block, along with the key. The result is a comprehensible text that is transmitted to the receiver. To retrieve the original text, the



Figure 2.1: Symmetric cryptography

decryption (D) block algorithm is used with the same key that was used for encrypting the initial text. If a different key is used, an output is generated, but it will be incorrect (and typically understandable).

The formulas used are as follows.

$$\begin{aligned}
 K_1 &= K_2 = K \\
 C &= \text{enc}(K, P) \quad \text{or} \quad C = \{P\}K \\
 P &= \text{dec}(K, C) = \text{enc}^{-1}(K, C)
 \end{aligned}$$

Note that $C = \{P\}K$ means "encrypt the plaintext P using the key K ".

The issue in the diagram (Figure 2.1) is represented by the dashed line: how can the key be securely shared between the sender and receiver? We'll see it in *Key distribution for symmetric cryptography*.

Symmetric algorithms

name	key (bit)	block (bit)	notes
DES	56	64	obsolete
3-DES	112	64	56...112-bit strength
3-DES	168	64	112-bit strength
IDEA	128	64	famous for PGP
RC2	8-1024	64	usually 64-bit key
Blowfish	32-448	64	usually 128-bit key
CAST-128	40-128	64	usually 128-bit key
RC5	0-2048	1-256	optimal when B=2W
AES	128-192-256	128	state-of-the-art

Some famous symmetric encryption algorithms (block)

There are many algorithms, and the table on the left represents just a small selection. The first column provides the name, the second indicates the key length, and the third specifies the basic unit each algorithm can encrypt. These algorithms are referred to as "**block algorithms**" because they operate on a fixed number of bits.

The **DES** algorithm, once a standard for many years, is now considered obsolete and should never be used.

The most commonly used algorithm of this kind is **AES**, currently recognized as the state of the art (the strongest).

RC5 performs optimally when the block size is double the word size of the CPU architecture (e.g., 64-bit architecture -> 128-bit block) on which the algorithm is implemented.

Why are there so many algorithms? Because there are various types of computers, and many algorithms are not suitable for low CPU power.

The EX-OR (XOR) function

It is the ideal "confusion" operator, available on all CPU.

The peculiarity of this truth table is that it has 50% of 0 and 50% of 1: if XOR is performed with 2 random inputs (probability 0:1 = 50% : 50%), then the output will also be equally random; while, for example, AND is more likely to produce 0. That means that XOR does not change the probability distribution of the input, even though it generates different outputs. Some properties:

- if $A \oplus B = Z$
then $Z \oplus B = A$ or $Z \oplus A = B$
- $A \oplus 0 = A$
 $A \oplus 1 = \bar{A}$
 $A \oplus A = 0$
 $A \oplus \bar{A} = 1$

\oplus	0	1
0	0	1
1	1	0

XOR function

DES

DES stands for "**Data Encryption Standard**" and it is a standard defined by *FIPS 46/2* (Federal Information Processing Standard, the body responsible for setting standards for the American government). The modes for applying DES to data that is not equally divided into blocks are mentioned in the FIPS 81 standard.

DES is a unique algorithm because it has a 64-bit key, but its effective strength is equivalent to that of a **56-bit key**, as 8 bits are used for parity. This means that when a key is generated for the DES algorithm, only 56 bits are truly random, and every 7 bits, the algorithm inserts a bit that serves as the parity of the preceding 7 bits. When an attacker attempts to crack DES, they only need to discover these 56 truly random bits. DES is the only algorithm that distinguishes between actual (bits used to create the key) and effective (total number of bits) bits.

Developed in the 1960s, DES uses a **64-bit data block**, a size chosen when computers were less powerful. To perform all the necessary mathematical computations, a special-purpose unit called an **encryption processor** was created because DES relies on it to perform:

- **XOR**: which is not a problem → elementary operation;
- **Shift**: not a problem → elementary operation;
- **Permutation**: expensive operation. The permutation is not random, there are several ones, but still implementing that was much more efficient if done directly in hardware.

Triple DES (3DES, TDES)

Triple DES is the repeated application of DES (three times). It is possible to use two or three different 56bit keys. Typically, the process involves taking the input, encrypting it, and then repeating this operation two more times, using the output of the previous encryption as the input for the next encryption (resulting in three consecutive encryptions).

Normally, it is implemented in **EDE** mode, which, in its standard form, requires two keys: the plaintext is first encrypted with key-1, then the decryption algorithm is applied to the output using key-2 (which does not actually decrypt but applies another transformation), and finally, this last output is encrypted again using key-1. This mode was chosen because setting $K_1 = K_2 = K_3$ allows for the implementation of the simple DES in EDE mode. Thus, both EEE and EDE work equally well. It has been shown that 3DES with two keys can have lower security guarantees than expected.

In particular, denoting K_{eq} as the equivalent key obtained by applying the discussed transformations:

- **3DES with 2 keys**, $K_{eq} = 56$ bits if $2^{59}B$ of memory is available, otherwise $K_{eq} = 112$ bits:

$$C' = \text{enc}(K_1, P) \quad C'' = \text{dec}(K_2, C') \quad C = \text{enc}(K_1, C'')$$

- **3DES with 3 keys**, $K_{eq} = 112$ bits:

$$C' = \text{enc}(K_1, P) \quad C'' = \text{dec}(K_2, C') \quad C = \text{enc}(K_3, C'')$$

The 3DES is a standard FIPS 46/3 and ANSI X9.52 (family X9 is standard for security in banking and financial applications).



Why not Double DES? The double application of any encryption algorithm is susceptible to a **known-plaintext** attack known as *meet-in-the-middle* (Section 2.2, see below), which allows for the decryption of data with at most 2^{N+1} attempts (if the keys are N bits long), instead of the 2^{2n} steps one would expect from an ideally secure algorithm with $2n$ bits of key. If the attacker knows one plaintext, they can execute the attack. For this reason, **the double version of encryption algorithms is never used** because the computation time doubles, but the effective key length increases by just one bit.

Furthermore, it has been proven that if the base symmetric algorithm is a group, then there exists an equivalent key K_3 such that:

$$\text{enc}(K_2, \text{enc}(K_1, P)) = \text{enc}(K_3, P)$$

This implies that in this case, the time needed for normal encryption/decryption doubles, but not a single bit is gained in terms of K_{eq} .

Meet-in-the-middle attack

hypothesis:

- N bit keys
- known P and C such that $C = \text{enc}(K_2, \text{enc}(K_1, P))$

note:

- $\exists M$ such that $M = \text{enc}(K_1, P)$ and $C = \text{enc}(K_2, M)$

actions:

- compute 2^N values $X_i = \text{enc}(K_i, P)$
- compute 2^N values $Y_j = \text{dec}(K_j, C)$
- search those values K_i and K_j such that $X_i = Y_j$
- "false positives" can be easily discarded if more than one (P, C) couple is available

By hypothesis, N-bit keys are used, and we have known plaintext (P) and ciphertext (C) such that $C = \text{enc}(K_2, \text{enc}(K_1, P))$. Note that $\exists M$ such that $M = \text{enc}(K_1, P)$, and $C = \text{enc}(K_2, M)$. The attacker computes 2^N values $X_i = \text{enc}(K_i, P)$ and then computes 2^N values $Y_j = \text{dec}(K_j, C)$. The search then involves finding those values K_i and K_j such that $X_i = Y_j$. There can be "false positives" but these can be easily discarded if more than one (P, C) pair is available.

Let's consider a company protecting its communication between Turin and Milan using double DES. If the attacker sniffs the network (and reads the encrypted data), they can send a message over that line (meaning they control the plaintext), and then observe the ciphertext (of the sent plaintext) automatically generated by the security system. This is a way to obtain a pair of (P, C) without knowing the keys.

2.2.1 Application of block algorithm

How is a block algorithm applied to a data quantity different from the algorithm's block size?
There are two cases:

1. Data size to encrypt > algorithm's block size

- **ECB (Electronic Code Book):** Nowadays considered not secure and must never be used.
- **CBC (Cipher Block Chaining):** Currently the best way to apply an encryption algorithm to data that is bigger than the size of the algorithm's block size.

2. Data size to encrypt < algorithm's block size

- **Padding:** Used only when the data size is not exactly a multiple of one block size (e.g., when data is 2.6 blocks long). Note that this technique is not used to pad data that is just shorter than one block size.
 - **CTS (CipherText Stealing):** Permits the use of block algorithms without padding, avoiding an increase in the size of the ciphertext compared to the plaintext.
- **CFB (Cipher FeedBack), OFB (Output Feedback), CTR (Counter mode):** These are application modes for an algorithm, not algorithms themselves.

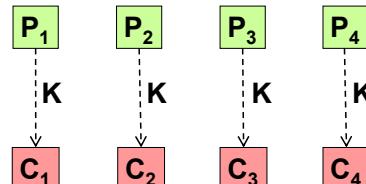
ATTENTION! These are NOT algorithms; they are application modes for an algorithm.

It is necessary to specify: the algorithm, the size of key, and if it is a block algorithm, also the mode of the application (e.g., AES-128-CBC).

ECB

Encryption This mode splits the plaintext into blocks, and each block is encrypted separately with the key; that means that, for each block $i = 0, \dots, N$,

$$C_i = \text{enc}(K, P_i)$$



Encryption with ECB

It must NOT be used! [Lioy said that the exam will not be passed if ECB is used somehow]. In particular, it must not be used on long messages (any message longer than 1 block) because:

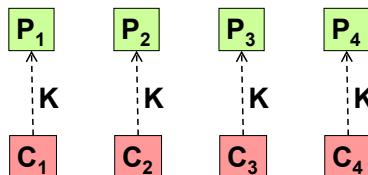
- If the attacker is intercepting the ciphertext and exchanges the position of 2 blocks, it is not detected (**block swapping**). This permits to exchange data inside an encrypted message;
- Identical blocks generate identical ciphertexts, hence it is vulnerable to **known-plaintext attacks**. Plaintext attacks require the precomputation of all possible encryptions of a known plaintext: comparing the encrypted block with all the precomputed possible encryptions, it is possible to figure out what the key is.

Known-plaintext attack example Let us suppose that we want to intercept a message sent by the rector of the Politecnico di Torino. It is possible to assume that his messages will contain the word “Torino”, so we start encrypting this word with all possible keys. Next, we can go sniffing the blocks sent by the rector: if any of these blocks is equal to the encryption of “Torino”, then we can use the key used to obtain that encryption of “Torino” to decrypt the rest of the message. Some arguments against this method could be: in which way is “Torino” written (e.g., “TORINO”, “Torino”)? What if “Torino”, since it is shorter than one machine word, is split into different blocks (e.g., $P_i = \dots To$, $P_{i+1} = rino\dots$)? The known-plaintext attack is made possible by the fact that usually people exchange data in a structured format, so that there are metadata that are usually known (e.g., fixed header).

If a Word file is created with just a word, it is big KBs of memory (and not just bytes). This happens because Word applies a header on the document which is always the same (and always at the beginning of a file). If the first block of a Word file is encrypted, it can serve as a dictionary for Word files. This means that we do not even have to guess some plaintext, but just the file format.

Decryption Decryption is a straightforward process as it involves taking the ciphertext block and decrypting it with the corresponding key. In the case of an error in transmission, it only affects the decryption of that specific block, and **there is no propagation of the error to other blocks**. For each block $i = 0, \dots, N$,

- $P_i = \text{enc}^{-1}(K, C_i)$



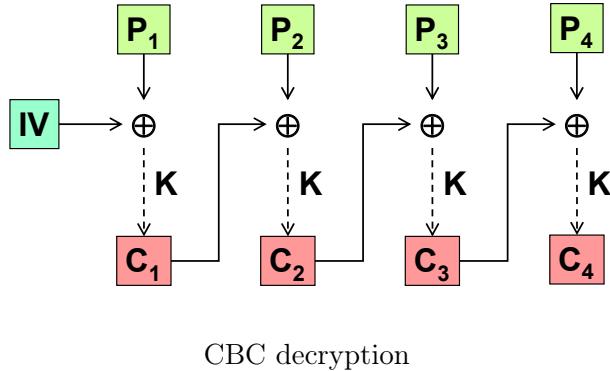
Decryption with ECB

CBC (Cipher Block Chaining)

Encryption This mode resolves all of ECB’s problems. It divides the plaintext into blocks, but before each block is encrypted, it is **XORed with the ciphertext of the previous block**. This introduces unpredictability in the encryption process. However, there is an issue with the first block (which serves as the header of the mentioned file); to apply CBC, an additional element called the **Initialization Vector (IV)** must be introduced, considered as C_0 , with the sole purpose of altering the first plaintext block.

CBC also provides protection against block swapping because if blocks are swapped, the output will be different due to being altered with the incorrect XOR element.

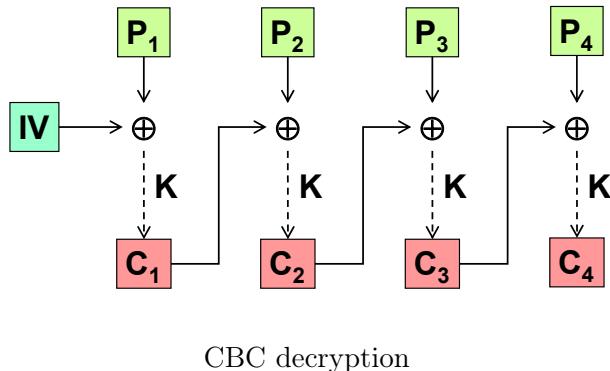
$$C_i = \text{enc}(K, P_i \oplus C_{i-1})$$



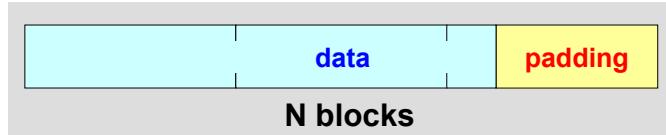
Decryption The IV vector needs to be known at the receiver as well because, when the ciphertext is received, the IV must be known to decrypt the first block (since XOR is the inverse operator of itself). The IV must also be different every time (hopefully random) to avoid being guessed, as its purpose is to make it impossible to precompute the possible encryptions of the first block. It must also be a **NONCE** (Number used once), meaning that the IV should be generated and never reused in the future.

In some cases, the IV is sent in clear because even if the attacker knows it, they could start the computation only at that moment, and this is a time-consuming task. Moreover, it will permit the attacker to attack only the sent message, as the next one will have another IV. The IV can also be sent encrypted using ECB since it is just a block.

$$P_i = \text{enc}^{-1}(K, C_i) \oplus C_{i-1}$$



Padding (aligning, filling)



It is not always possible to have the plaintext split into blocks that are all the same size. In the picture, the data consists of two complete blocks and a partial block at the end. To encrypt this small part, a technique called padding is used. Bits are added at the end of the original data until it reaches a multiple of a block. However, it has some problems since:

- The ciphertext gets longer than the plaintext: adding useless data means transmitting / storing more data than needed;
- Which should be the value of padding bits? How can we distinguish the padding from the original plaintext at the receiver?

Padding Techniques:

- If the length is known or it can be obtained as in the case of a C string, it is possible to add null bytes at the end: 0x00 0x00 0x00;
- The original DES specified to add one "1" bit followed by as many "0" as needed: 1000000;
- One byte with value 128 followed by null bytes: 0x80 0x00 0x00;
- Last byte's value equal to the length of padding: 0x?? 0x?? 0x0 (3 bytes of padding. The last one has value 3, and the previous has a value to be specified).

There are numerous padding techniques available, and the choice of a specific padding method plays a crucial role in preventing or mitigating various types of attacks.

There are many kinds of values that can be selected (*do not remember all the techniques, just remember there are many and the proper one must be selected*):

- (Schneier) null bytes, e.g., 0x00 0x00 0x03
- (SSL/TLS) bytes with value L , e.g., 0x03 0x03 0x03
- (SSH2) random bytes, e.g., 0x05 0xF2 0x03
- (IPsec/ESP) progressive number, e.g., 0x01 0x02 0x03
- Byte with value $L - 1$, e.g., 0x02 0x02 0x02

Some notes

- B stands for the size of the algorithm's block
- D stands for the size of data to process

Some of these techniques offer (minimal) integrity control: if the key is wrong or data is manipulated, then the padding bytes are incoherent (e.g., length of something bigger than one block or wrong padding values).

Typically, this is applied to large data, on the last fragment resulting from the division into blocks (e.g., for ECB or CBC). If $|D| < |B|$, an ad-hoc technique is preferred (CFB, OFB, CTR, ...). If there is just a byte, it is not good to add 65 bytes of padding.

Even if the plaintext is an exact multiple of the block, padding must be added anyhow to avoid errors in the interpretation of the last block. The biggest padding is required when there is no padding to add.

With SSH2, padding equal to data gives different ciphertexts (even when encrypted with the same key).

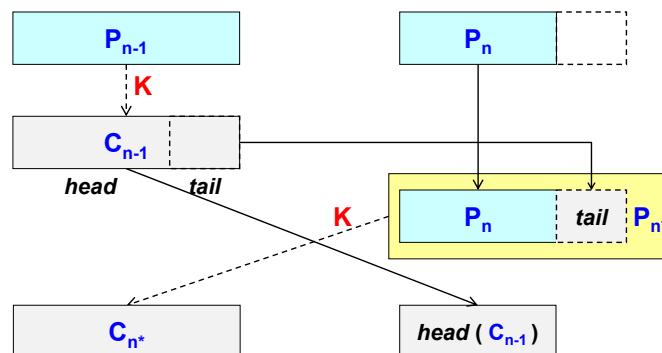
The padding type for a certain algorithm determines the type of (some) possible attacks, but it also depends upon the algorithm used.

Ciphertext stealing (CTS)

When using padding, the size of the ciphertext is bigger than the plaintext. This may not be acceptable in several cases (for example, data encrypted on a hard disk that may not fit on the same disk). It is possible to use CTS instead of padding. The last (partial) block is filled with bytes taken from the second-to-last block (encrypted), then these bytes are removed from the second-to-last block (which becomes a partial one). After encryption, the positions of the last and second-to-last blocks are exchanged.

It is useful when it is not possible to increase the size of data after encryption, but the computation time slightly increases.

CTS Example with ECB (encryption)



1. The plaintext is split into blocks;
2. The second-to-last block is full, so it is possible to encrypt it with the key;
3. The last one is not full and creates a problem;
4. The encrypted block C_{n-1} is split into two parts: *head* and *tail*. The tail has the same size as the part that is missing in the last block;
5. The tail is added to the partial block, creating a full block;
6. The fully created block is transmitted, and then the head is transmitted.

Chapter 3

Authentication techniques and architectures

3.1 What is authentication

Definitions of authentication

There are three different definitions of *authentication*:

- **RGC-4949 (Internet security glossary):**
"the process of verifying a claim that a system entity or system resource has a certain attribute value"
- **whatis.com:**
"the process of determining whether someone or something is who or what it is declared to be"
- **NIST IR 7298 (Glossary of Key Information Security Terms):**
"verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system"

The key aspects of these definitions are that they define the authentication of an actor, meaning that **it could be not only a human being** (interacting via software running on hardware) **but also a software component or a hardware element** (interacting via software). The common shorthand for authentication is `authN` or `authC`, while `authZ` is used for authorization, *which is different but related*.

Authentication factors

While authenticating an *actor*, there are three categories of **authentication factors** that can be used:

- **Knowledge:** authentication relies on something that *only the user knows*, for example a static passphrase, code, or personal identification number.

The associated risks involve the storage of this knowledge, how it can be demonstrated, and the way it is transmitted.

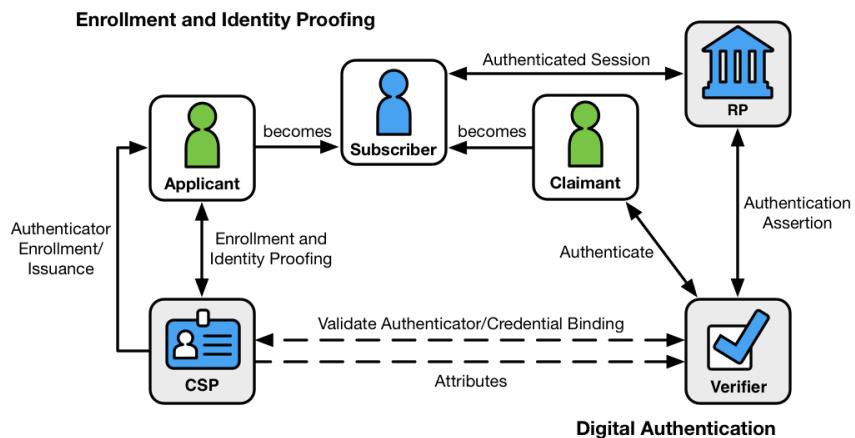
- **Ownership:** authentication relies on something that *only the user possesses* (often called an "authenticator"), for example, a token, smart card, or smartphone.

The associated risks can be related to the authenticator itself, such as the possibility of infection with malware, the potential for it to be manufactured in a country that imposes government control, or the risk of it being stolen, cloned, or used without the owner's authorization (e.g., forgetting an unlocked smartphone).

- **Inherence:** *something the user is*, for example, a biometric characteristic (such as a fingerprint).

The associated risks include counterfeiting and privacy concerns. Inherence factors pose a greater risk than the previous cases because, for example, a biometric characteristic cannot be replaced when compromised. For this reason, inherence factors should be limited to very secure environments, typically used only for local authentication, as a mechanism to unlock a secret or a device.

3.2 Digital authentication model (NIST SP800.63B)



General model for digital authentication as described in NIST SP800.63B

- In this model, an actor who wants to use a system is called an **applicant**: if it possesses an authenticator it can provide it to the **CSP (Credential Service Provider)**, or it can get one (for example, when a student is enrolled in Politecnico, he is given a smart card that works as an authenticator). The CSP is that component that will issue or enrol user credential and authenticator, and verify and store associated attributes.

When this procedure is completed successfully, the actor becomes a **subscriber**, which is an entity recorded in the authentication system.

- Later, when the actor wants to use some network service, typically the actor is called a **claimant**, because they claim to be a valid user. Typically, an authentication protocol against a **verifier** is run to verify this claim. When this process ends successfully, the actor becomes **a subscriber with an open authenticated session** with the **relying party**, that will request and receive an authN assertion from the Verifier to assess user identity (and attributes).

The relying party, which requests the actor to be authenticated, is the end application. The verifier may have communication with the CSP to validate the binding between the authenticator used in the authentication protocol and the credential claimed.

Recap

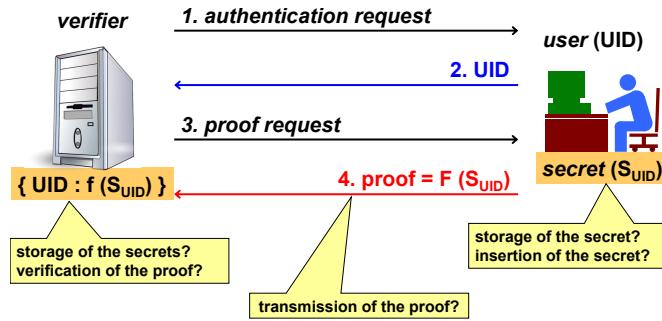
- The credential binds an authenticator to the subscriber via an ID:**
 - for example, an X.509 certificate can be considered the credential, as it binds the identity and attributes written inside the certificate with the authenticator. In this case, the authenticator is the private key that the user controls.
- CSP (Credential Service Provider):**
 - will issue or enrol user credential and authenticator
 - verify and store associated attributes
- Verifier:**
 - executes an authN protocol to verify possess of a valid authenticator and credential
- Relying party:**
 - will request/receive an authN assertion from the Verifier to assess user identity (and attributes)

These roles may be separate or collapsed together. Thinking about a *Linux machine* used locally, the enrollment phase involves creating a new user with a username (the credential) and a password (the authenticator). In this scenario, the CSP is the operating system itself, and when a user wants to use a server on this machine, they need to perform login, which is the verifier. The relying party is any software running on that machine that uses the identity as proven by the login service of the operating system.

Another example is the use of Google Identity for different services, such as the Doodle Service to agree on a date. For Doodle, there is the option to use Google or Facebook Credentials. In this case, the relying party is Doodle, while the verifier (as well as the CSP) is Google or Facebook.

Another example is the use of Google Identity for different services, such as the Doodle Service to schedule an appointment. For Doodle, there is an option to use Google or Facebook Credentials. In this case, the relying party is Doodle, while the verifier (as well as the CSP) is Google or Facebook.

3.3 Generic authentication protocol



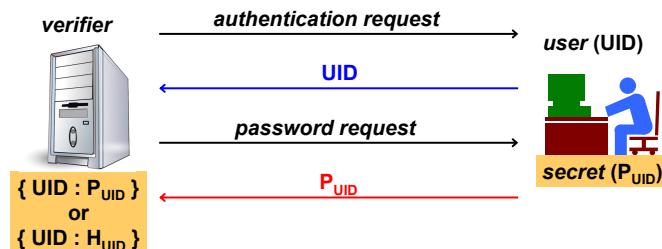
Suppose that the user wants to access an application server (Relying Party). In this case, the server will include both the Relying Party and Verifier. The user has been identified with the User ID and has a secret associated with that User ID. The server's identifier contains a table with the User ID and the result of the function f applied to the secret. Normally, the secret should never be stored in cleartext; of course, if the function f is the identity function, it means you are storing the secret in cleartext, which is not recommended.

When the user wants to access the service, they receive an **authentication request**. Initially, they provide the **User ID**, and then the verifier asks for a **proof request**; the user replies with the **proof = $F(S_{\text{UID}})$** , which is the result of the computation with the function F applied to the user's secret.

In this scenario, several problems need to be addressed:

- On the user side, how is the secret stored? How is the secret provided (e.g., if it is a password entered via a keyboard, a keylogger could disclose it)? Is the transmission of the proof secure?
- On the server side, how are the secrets related to the user stored? When a proof is received, how is it verified to be the correct proof?

3.4 Passwords (reusable)



Imagine that the secret is a **reusable password** (meaning that it is always the same), and the user is identified by their User ID, with the secret being the password associated with that

user. On the server side, there is a table containing usernames and passwords in plain text or a function H computed over the password.

Once again, there will be an **authentication request**, followed by the user sending their **UID**. Then, a password request is made, and the user responds with P_{UID} .

Assume the network is secure and focus on the verifier's side. The secret is the user's password, and the client creates and transmits the proof, typically using a function $F = I$, which is the identity function. The proof is the password sent in plain text, which is, of course, dangerous.

Now, looking at the server, when the server receives the password, it needs to check if it is correct or not:

1. first case: if the function f used to store the password is the **identity function** ($f = I$), then the proof is the password in cleartext. In this scenario, the server knows all the passwords in plain text, and verifying their correctness is simple. However, it is risky; if someone copies the database, they will gain access to all data.
2. second case (the suggested one): f is not the identity function but a **one-way hash** (a digest of the password), and the server does not know the password in plain text but only the (unprotected) digest H_{UID} . This means that access control is a bit more complex, as when the proof is received, the hash of the proof is computed and compared with the hash stored in the password database. If the database is stolen, the attacker will not have a copy of the plain-text passwords.

Problems of reusable passwords

Password-based authentication is usually convenient for the user, but only if they have to remember just one password, a reusable one. The current situation is unfortunate because in some applications, there is a need for several passwords that cannot be remembered by a person, so they would need to be stored on the user's side, which is a source of insecurity. The **disadvantages** of password-based authentication are: The disadvantages of password-based authentication are:

- The user-side password storage: it could be written on a post-it or on a client-side password manager (also called password wallet), that stores it encrypted typically using only one passphrase;
- Guessable passwords;
- Server-side password storage: the server must know the password in cleartext or an unprotected digest of it (dictionary attack);
- **Sniffing**: Password can be sniffed while it is sent across the network;
- **DB attacks**: There could be attacks to the password DB at the verifier (if DB contains plaintext or only obfuscated password);
- **Password guessing**: it is very dangerous if it can be done offline, for example against a list of password hashes;

- **Password enumeration:** if the password is limited in length or character type, or if authN protocol doesn't block
- **Password duplication:** using the same password for different services, due to user password reuse. This could be a problem because if the user has the same password for a high-security service and for a weaker one, an attacker could discover it on the weaker system and have access to the high-security one;
- **Cryptography aging:** the solution adopted for verifying the secret should not be tied to a specific cryptographic algorithm, because it could be then difficult to adapt to the need for changing the algorithm used, due to new attacks and more computing power;
- **Password capture** via server spoofing and phishing;
- **MITM attacks.**

Password best practice

- Use a **mixture** of alphabetic characters (both uppercase and lowercase), digits, and special characters. Unfortunately, there are many systems that don't allow the use of special characters or impose limits on password length.
- Use a **long** password, preferably at least 8 characters in length.
- **Avoid using dictionary words**, as attackers often employ dictionaries from multiple languages.
- **Change your password frequently.** If the same password is kept for an extended period, attackers have more time to perform their computations. It's advisable to change your password at least once or twice a year to reduce the window of exposure.
- Whenever possible, **consider not using passwords**. However, this may be unavoidable unless biometric techniques are employed.

Storing passwords

Storing passwords on the server-side

- Never store passwords in cleartext.
- If the password is **encrypted**, the server must have access to the encryption key in cleartext, which can be a security concern. To enhance security, it's recommended to **store a password digest**. However, be cautious of dictionary attacks that can be expedited by techniques such as **rainbow tables**. To mitigate these types of attacks, you can introduce an unpredictable element known as a "**salt**".

Storing password on the client-side

- Should be only in user's head;
- If passwords are a lot, use an encrypted file or a password wallet;
- It's better use an encrypted file, or a "password wallet / manager".

Storing passwords on the client-side

- Passwords should be memorized by the user.
- If there are numerous passwords to manage, consider using an encrypted file or a password wallet.

3.4.1 The "Dictionary" Attack



If you store the plain hash of a password, dictionary attacks are possible. This is possible under two hypotheses:

1. known hash algorithm;
2. leakage of information, so that the attacker has a copy of the **password hash values**.

Hashes are not invertible functions, but it is possible to make a **pre-computation**. Therefore, even if there is no copy of any password hash yet, it is possible to decide that it would be worthwhile to attack passwords stored as plain SHA-1 hashes in the future.

You must obtain a dictionary containing not only the Italian language but all possible languages. For each word in the dictionary, you compute the hash of the word and store it in a database paired with the corresponding word. By "word," we mean a potential passphrase, not a part of it. Typically, attackers have dictionaries extended to include words such as names of famous people.

The main hypothesis is that the user has chosen one of the words contained in the dictionary. The attack proceeds as follows:

1. At some point, the attacker obtains a hash value due to a leakage.
2. The attacker performs a simple **lookup** as follows: $w = \text{lookup}(\text{DB}, \text{HP})$, where DB is the database and HP is the computed hash of a word in the database, if any of the hash passwords appears in any tuple.

3. If the response is positive, the password is equal to that word. If not, the password is not from the dictionary.

Pre-computation is the key because if you wait until you get a copy of the password hash, and only at that point you start computing all the possible hash values, it could be too late because the password could have changed.

3.4.2 Rainbow table

A dictionary attack can be made faster and more effective by the **Rainbow table** technique. It is still a *dictionary attack*, but it involves a trade-off between space and time. Trying all possible passwords and computing the hash would be fast, but the result would be a huge database. If you have a complete database, the lookup would be fast, but fewer passwords are stored, and a bit more time is taken to compute the password if the corresponding hash is present. This is an improvement because it makes an exhaustive attack feasible for certain password sets.

Imagine creating a rainbow table to attack a password that we know contains 12 digits. The exhaustive attack would require 10^{12} rows, which is a huge number of lines containing passwords and the corresponding hash values. A rainbow table could be used to reduce the number of rows in the database by a factor of 1000. In this way, we get a 10^9 rows database, where each line represents 1000 passwords. To achieve that, we use the **reduction function**:

$$r : h \Rightarrow p$$

It is a function r that takes a hash as input and creates one possible password. **Beware that this is NOT the inverse of the hash (h^{-1})**, because the inverse of the hash does not exist. It is just a mapping function that, from a hash, creates one of the possible passwords of the whole set; in other words, the reduction function is a different function with a swapped domain and codomain of the hash function.

For other informations, check https://en.wikipedia.org/wiki/Rainbow_table.

Pre-computation Then, the pre-computing is the following:

```
■ pre-computation:  
■ for ( 109 distinct P )  
■ for (p=P, n=0; n<1000; n++)  
    - k = h(p);  p = r(k);  
■ store ( DB, P, p ) // chain head and tail
```

1. Select 10^9 distinct passwords (the desired size) called P .
2. For each of them, initialize the computation starting from that specific password, and then iterate 1000 times; each time the hash of the current password is computed (called k), and then the reduction function is used to go from k to another possible password.

3. At the end, the password P of the first cycle is stored in the database together with the last computation of the reduction function (called p).
4. Then, the entry implicitly represents all the 1000 passwords tried. Note that there is no more hash to be stored.

Attack Then the attack rises in this way:

```
■ attack:  
■ let HP be the hash of a password  
■ for (k=HP; n=0; n<1000; n++)  
    ■ p = r(k)  
    ■ if lookup( DB, x, p ) then exit ( "chain found!" )  
    ■ k = h(p)  
■ exit ( "HP is not in any chain of mine" )
```

1. HP is the leaked hash of a password;
2. Start an iteration of at most 1000 times, and each time, the reduction function is used to derive a possible password from the hash value;
3. Next, search the database to check if there is a row where p (the result of the reduction function) is at the end of the chain. In that case, we found the chain containing that hash; otherwise, a new value k is calculated by performing the password hash;
4. After finding the chain, the computation of the hash must be done again to identify which hash matches the one we have.

The problem is that since the reduction function is going from a hash to one possible password, there could be two different hashes that generates the same password, and this is called **fusion**. Rather than using a reduction function, a set of n reduction functions is used, one for each reduction step. On internet there are on sale pre-computed rainbow tables for various hash functions and password sets (e.g., SHA1 for alphanumeric).

This technique is used by various attack programs.

3.4.3 Using salt in storing passwords

The critical point in the previous kind of attack is that the attacker performs pre-computation. Without the rainbow table and without the database created by the dictionary, it would take a lot of time. For this reason, **do not provide the attacker with the information needed for pre-computation**, because it is based on the idea that the attacker may know which is the password (through the dictionary).

- Using the following technique, even if it can be possible to guess what a possible password is, the attacker does not get the hash table because every time a User ID is created, the

system generates a **salt** that is different for each user.

The salt is a random (unpredictable) and long (increased dictionary complexity) string of bytes. Users do not have to memorize the salt, which should contain rarely used or control characters.

Then the hash is computed using the password concatenated with salt:

$$HP = \text{hash}(\text{pwd}|\text{salt});$$

- The verifier stores UID, HP_{UID} and salt_{UID} .

If someone gets the information in the database, he also gets the salt, but only then the computation can start, which will require a lot of time and in the meanwhile the password could have been changed. Additionally, there are different HP for users having the same password.

This makes the dictionary attacks nearly impossible (including those based on rainbow tables).

Example: Passwords in Linux

Originally stored in `/etc/passwd`, hashed with a DES-based hash function named `crypt()`. Since `/etc/passwd` needs to be world-readable (contains usernames, UID, GID, home, shell, etc.), passwords have been moved to `/etc/shadow`, readable only by system processes. Passwords are stored in the following form - see `crypt(5)`:

```
$id$salt$hashedpwd
```

Different hash functions are used depending on the ID, for example:

- 1 = MD5, ..., 5 = SHA-256, 6 = SHA-512, ...

If `idsalt` is absent, the old DES-based hash is used (with a 12-bit salt, and the password is truncated to 8 characters) - danger! Some algorithms have adjustable complexity (to counter brute-force attacks).

Case history: The LinkedIn Attack

In June 2012, someone was able to copy 6.5 million passwords from LinkedIn, which were unsalted plain SHA-1 hashes. The person published those hashes on the internet and asked for crowdsourcing, used for cooperative password cracking (which means trying to compute SHA-1 hashing of words and looking if someone has a match). At least 236,578 passwords were found before Interpol was able to ban the website that published the password hashes.

Note that simultaneously LinkedIn found out that the LinkedIn app for iPad/iPhone was sending in clear sensitive data (not relevant to LinkedIn!).

Example: Passwords in MySQL

MySQL is a database where usernames and passwords are stored in the "user" table. MySQL (from v4.1) uses a **double hash (without salt!)** to store passwords:

```
SHA-1(SHA-1(password))
```

Then, the hex encoding of the result is stored, preceded by * (to distinguish this case from MySQL versions < 4.1). For example, for the password "Superman!!!," the field `user.password` is

```
user.password = *868E8E4F0E782EAA610A67B01E63EF04817F60005
```

To verify that this is the double hash of the word, you can use the following command on Linux:

```
$ echo -n "Superman!!!" | sha1sum | xxd -r -p | sha1sum
```

This is the standard way for MySQL to store passwords, which is not secure. It is advisable to change the standard way MySQL uses to store passwords by using a salted approach.

3.5 Strong (peer) authN

Recently, there has been a growing emphasis on the requirement to move away from standard authentication methods and adopt strong peer authentication. While this is consistently requested in specifications, it is often not formally defined or defined in multiple, potentially confusing ways.

3.5.1 ECB Definition for Internet Banking

According to the European Central Bank (ECB), strong customer authentication (authN) is a procedure based on the use of two or more of knowledge, ownership, and inherence. The selected elements must be mutually independent, ensuring that the breach of one does not compromise the others. At least one element should be non-reusable and non-replicable (except for inherence), and not capable of being surreptitiously stolen via the Internet. The strong authentication procedure should be designed to protect the confidentiality of the authentication data; for example, if a password is used, it cannot be sent in clear text.

3.5.2 PCI-DSS definition for payment with credit cards

According to PCI-DSS definition, which is for payment with credit cards, starting from v3.2 it requires multi-factor authentication (MFA) for access into the cardholder data environment (CDE): it does not matter if it a trusted or untrusted network, and it is also used when the access is performed by administrators. The only exception is with direct console access (physical security), which means that you enter the room where the server is placed. For remote access it is always required from untrusted network and by users and third-parties (such as maintenance). This was the best practise until Jan '18 and it has been made compulsory afterwards. **Remember:** MFA is **not** twice the same factor (e.g., two passwords).

3.5.3 PCI-DSS Definition for Payment with Credit Cards

According to the PCI-DSS definition, which applies to payment with credit cards, starting from v3.2, multi-factor authentication (MFA) is required for access into the cardholder data environment (CDE). This requirement is applicable regardless of whether the network is trusted or untrusted, and it is also mandatory for administrators accessing the CDE.

The only exception is for direct console access, which involves physical security measures, such as entering the room where the server is located. However, for remote access, MFA is always required, especially from untrusted networks, and for users and third parties (e.g., maintenance).

This best practice was in effect until January '18 and became mandatory thereafter.

Remember: MFA does **not** mean using the same factor twice, like using two passwords.

3.5.4 Other definitions

According to the *Handbook of Applied Cryptography*,

authentication is a **cryptographic challenge-response identification protocol**.

More in general, it is a technique resisting to a well-defined set of attacks.

An authN technique can be classified as strong or weak depending on the attack model:

- E.g., users of Internet Banking follow the ECB definition;
- E.g., employees of PSP adhere to the PCI-DSS definition.

In general, pay attention to your specific application field because it defines the types of risks and the level of strength required for your strong authentication.

3.6 Challenge-Response Authentication (CRA)

Challenge-response protocol is a possible way to implement strong authentication. CRA means that there is a challenge sent to the Claimant from the Verifier. The Claimant replies with the solution computed using some secret knowledge and the challenge. The Verifier compares the response with a solution computed via a secret associated with the Claimant.

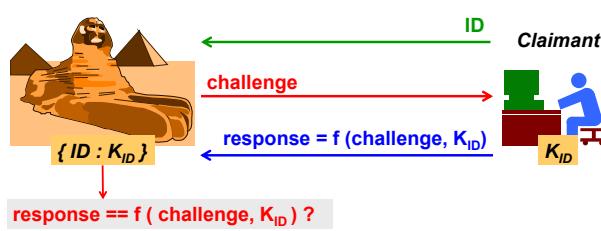


Someone claims to own the identifier (ID). The verifier looks that there is a row associated with that ID and sends to the claimant the challenge. The claimant has a key (K_c) and uses it to perform some kind of computation (function f) and generates a response. The response can be checked by applying the function g to the challenge and to a well-known key (K_v) of the Verifier. The keys can be different or the same.

General issues of CRA

- The challenge must be non-repeatable to avoid replay attacks. For this reason, usually, the challenge is a (random) nonce.
- The function f must be non-invertible, otherwise, a listener can record the traffic and easily find the shared secret by using the function $K_c = f^{-1}(response, challenge)$.

Symmetric CRA



In this case, there is a common key shared between Claimant and Verifier, which is typically the password or passphrase of the user. The function f is computed two times: once from the user to make the response, and once from the verifier to verify the match.

General issues of Symmetric CRA

General issues with Symmetric CRA are:

- The easiest implementation uses a hash function (faster than encryption) such as SHA1 (deprecated), SHA2 (recommended) or SHA3 (future);
- K_{ID} must be known in cleartext to the Verifier and this may lead to attacks against the $ID : K_{ID}$ table at the Verifier;

Chapter 4

Security of IP networks

Chapter 5

Security of network applications

Chapter 6

Firewall and IDS/ISP

Chapter 7

E-mail security

Appendix A

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<<https://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any

member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is

precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year

after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License

and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <https://www.gnu.org/licenses/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit

those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.