# Computer Security

Matteo Secco

February 27, 2021

# Contents

# 1 Introduction to Computer Security

## 1.1 Security requirements

### CIA Paradighm

**Confidentiality** Information can be accessed only by authorized entities

**Integrity** information can be modified only by authorized entities, and only how they're entitled to do

**Availability** information must be available to entitled entities, within specified time constraints

The engineering problem is that **A** conflicts with **C** and **I**

# 2 Computer Security Concepts

## 2.1 General concepts

**Vulnerability** Something that allows to violate some CIA constraints

- The physical behaviour of pins in a lock
- A software vulnerable to SQL injecton

**Exploit** A specific way to use one or more vulnerability to violate the constraints

- lockpicking
- the strings to use for SQL injection

**Assets** what is valuable/needs to be protected

- hardware
- software
- data
- reputation

**Thread** potential violation of the CIA

- DoS
- data break

**Attack** an intentional use of one or more exploits aiming to compromise the CIA

- Picking a lock to enter a building
- Sending a string creafted for SQL injection

**Thread agent** whoever/whatever may cause an attack to occour

- a thief
- an hacker
- malicious software

**Hackers, attackers, and so on**

**Hacker** Someone proficient in computers and networks

**Black hat** Malicious hacker

**White hat** Security professional

**Risk** statistical and economical evaluation of the exposure to damage because of vulneravilities and threads
$Risk = \underbrace{Assets \times Vulnerabilities}_{\text{controllable}} \times \underbrace{Threads}_{\text{independent}}$

**Security** balance of (vulnerability reduction+damage containment) vs cost

## 2.2 Security vs Cost

**Direct cost**

- Management
- Operational
- Equipment

**Indirect cost**

- Less usability
- Less performance
- Less privacy

**Trust** We must **assume** something as secure

- the installed software?
- our code?
- the compiler?
- the OS?
- the hardware?

# 3 Introduction to crypthography

**Kerchoffs' Principle** The security of a (good) cryptosystem relies only on the security of the key, never on the secrecy of the algorithm

## 3.1 Perfect Chipher

- $P(M = m)$ probability of observing message m
- $P(M = m|C = c)$ probability that the message was m given the observed cyphertext c

**Perfect cypher:** $P(M = m|C = c) = P(M = m)$

**Shannon's theorem** in a perfect cipher $|K| \geq |M|$

**One Time Pad** a real example of perfect chipher

**Algorithm 1** One Time Pad

```
Require: $len(m) = len(k)$
Require: keys not to be reused
  return $k \oplus m$
```

**Brute Force** perfect chyphers are immune to brute force (as many ”reasonable” messages will be produced). Real world chiphers are not.
A real chipher is vulnerable if there is a way to break it that is faster then brute forcing

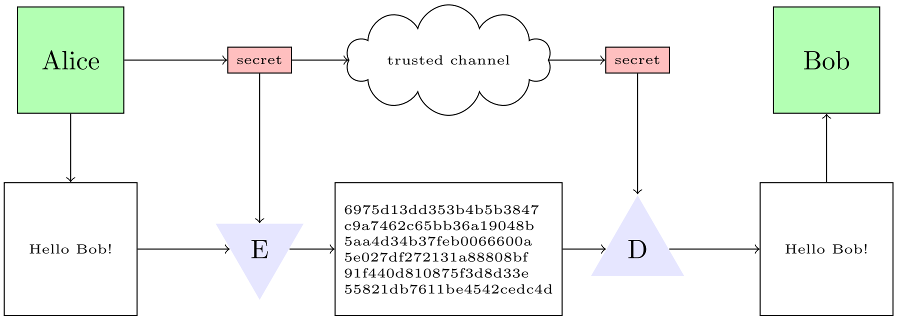
**Types of attack**

**Ciphertext attack** analyst has only the chipheertexts

**Known plaintext attack** analyst has some pairs of plaintext-chiphertext

**Chosen plaintext attack** analyst can choose plaintexts and obtain their respective ciphertext

## 3.2 Symmetric encryption



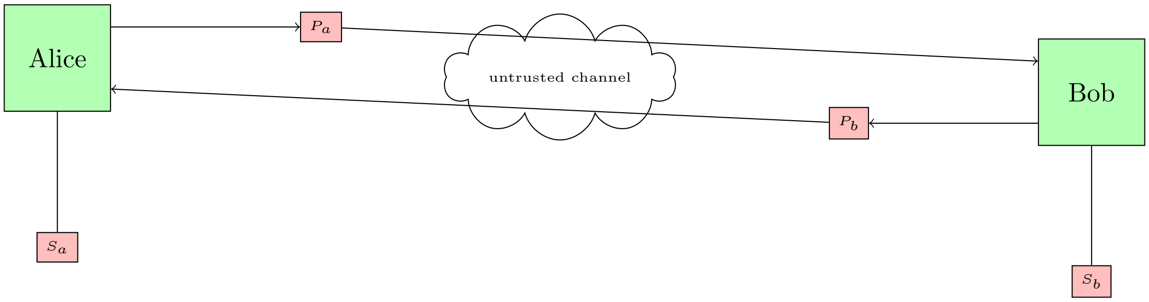
Use **K** to both encrypt and decript the message
Scalability issue
Key agreement issue

### 3.2.1 Ingredients

**Substitution** Replace each byte with another (ex: caesar chipher)

**Transposition** swap the values of given bits (ex: read vertically)

## 3.3 Asymetric encryption



Each user owns a private and a public key $(S_i, P_i)$, where the public key is publicly available. The cryptoalgorithm is designed so that messages encrypted using $P_i$ can only be decrypted using $S_i$. This allows Alice to encrypt a message using $P_{bob}$, and Bob (and nobody else) to decrypt is using $S_{bob}$. Also, to prove its identity, Bob could send a message encrypted using $P_{bob}$. When Alice manages to decrypt is using $P_{bob}$, she can be sure that the message came from Bob

## 3.4 Hash functions

A function $H : X \to Y$ having $|X| = \infty$ but $|Y| = k \in \mathbb{N}$. This means $|Y| < |X|$, leading to <u>collisions</u>: couples $x_1, x_2 \in X : H(x_1) = H(x_2)$.

**Safery properties** are proberties needed to ensure robustness of $H$. In particular, it must be computationally infeasible to find:

**preimage attack resistance** $x : H(x) = h$ with $h$ known/crafted

**second preimage attack resistance** $y : y \neq x \wedge H(x) = H(y)$, where $x$ is known/crafted

**collision resistance** $x, y : H(x) = H(y)$

### 3.4.1 Attacks to Hash Functions

**Preimage attack** Given an hash $h$, the attacker can find $x$ such that $H(x) = h$, or given $x$, they can find $y$ such that $H(x) = H(y)$. This can be done faster than brute force.
With $|Y| = n$, random collisions happen in $2^{n-1}$ cases

**Simplified collision attack** The attacker can generate $x, y : \ H(x) = H(y)$ faster than brute force.
Random collisions happen in $2^{n/2}$ cases (for the Birthday paradox)

## 3.5 Digital Signature

To digitally sign a message, we first hash the message. Then, we encrypt the hash with our private key.
This however only guarantees that the sign was produced using our secret key, but someone may have stolen/guessed our private key.

### 3.5.1 PKI

Public Key Infrastructures is a service entitled to associate an identity to a key. To do so it uses a trusted third party called **Certification Authority**. The CA signs files called **digital certificates**, which bind an identity to a public key.

**Top-level CA** is a special CA that self-signs its certificates. It is a trusted element. The Root CA can then sign certificates for other CAs. In practice, a Root CA is a real world CA (the state, a regulatory organization...)

**Revocation** Signatures cannot be revoked, but certificates can be revoked (declared invalid), for example because the private key has been broken.
To do so, a Certificate Revocation List must exist for each CA