# 6.033 Lecture 20
# Security I

Americo De Filippo

July 6, 2023

## References

[1] Saltzer, Jerome H. and M. Frans Kaashoek. Principles of Computer System Design: An Introduction (2009): **Section: 11.1**

The words protection and security will be used as synonymus in the course. Any system has some policy about what data should be access from some people. What we are gonna talk about is mechanics that we are going to use in order to protect data.

**Vs Real World** Some of the goal are similar to those in the real world: encrypt data (lock on a door), what is legal (laws). The main differences between the 2 field are the change into the time of the teck $\frac{dtech}{dt}$, very fast attach and cheaper, in the end the laws are not easy to make it respect.

## 1 Negative goals

A really often goals for security systems are like: "Sam shouln't access this file" instead of: "Sam can access this file" (easy). The first thing is not easy to make it respect, because I have many ways in order to access a file. This is one of the thing that makes hard building security systems.

## 2 Client Server model

There are a list of thing that want to do as a server in order to give the access to same client: Let's say that Alice is our Client, Bob is our server, Eve is the ones that listen and take some information that should have and Lucifer that modify the packets in a malicius way.

- **Authenticate**: in order to give access only to Alice and not to Lucifer.
- **Authorize**: we have do understead if Alice can access some file.

- **Keep confidential**: Eve can tell the content the packet that hear.
- **Accountability**: is always possible that something could go wrong.
- **Avaibility**: Lucifer cannot do ddos attacks.

### 2.1 Safety net approch

The safety net approch abdicate some way to think about your system.

1. **Be paranoid**: feedbacks, defend in depth, minimize what is trusted.
2. **Consider environment**: The position of the server, the connections that are running on it, the people that can access the computer etc...
3. **Plan for iteration**: Assume that there will be security problems, assure that if a part of your system is being violeted the other parts remain safe.
4. **Keep audit trails**: Keep track that all the authetication and all the request keep tract of what they did etc... just keep track.

Most of the problems are caused by humans problems, let's say that I for mistake give my password to someone else, from that moment on for the system there are not violation when in fact there are. So you should be think about the UI, good defaults, least privilege.

## 3 Layers in security

At the top we have some application that we want to secure. Underneath we have 3 layes:

1. **Functionaliy**: Authenticate, authorize, confidetiality.
2. **Primitive**: sing very, ACL (access control list), encrypt decrypt.
3. **Cryptography**: cyptographic cypers, hashes.

### 3.1 The Cryptography layer

Early cryptography reliead on the idea to keep the protocol secret. Like A maps to B, keeping this protocol secret you can have a encypted message.

#### 3.1.1 Close design cypto scheme

This is based on the idea of making the protocol secret to anyone that is out of the range. This turns out to do not be a good idea, cause if someone discover the protocol you will have blow out everything into the system.

### 3.1.2 Open design systems

The idea is that we are going to have known algoritms and a secret key. We also have public keys that relies on mathematical function that does something with our private key and the public one.

### 3.1.3 Shared key system: One-time Pad

This is the example of a protocol that is impossible to break with some mathematical attack (perhaps in some other means). This relie on generating bits completely at random (which is not easy), also they are going to be a lot. Let's say that alice write a long random string of bits, it does the XOR with (bit-wise) with the known keys. So what happens is that the message that flow onto the network after the XOR looks like a random bits stream. On the other side will be done the inverse operation and decypt the message.

### 3.1.4 RSA Public keys protocl