So far we have learnt about:
- Controlling access to objects from security domain
- What about people’s access to the computer

Today we will talk about Authentication.

**The goal of security:**
- Map user to domain
- Identify user
- Prove user is who they claim to be (authentication)

Thinking about the scenario in below picture:

![Diagram showing Terminal and Server connected by a wire](image)

What can be happened?

**Threats:**
- Wiretapping
  - Sniffing
  - Spoofing
- Terminal Compromise
  - Key logging
  - Other covert channels
  - Spoofing
- Server attacks
  - Attacker may break into server
Three kinds of authentication:

- Something you know
- Something you have
- Something you are

The simplest authentication scenario:

```
<table>
<thead>
<tr>
<th>Terminal</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rob</td>
<td>Name</td>
</tr>
<tr>
<td>foobar</td>
<td>Passwd</td>
</tr>
<tr>
<td></td>
<td>rob</td>
</tr>
<tr>
<td></td>
<td>foobar</td>
</tr>
</tbody>
</table>
```

The problems in this scenario:

- Password is stored in clear on server
- Password is sent in clear
- Terminal can steal password
- Human factors in passwords:
  - Easy to guess
  - Automatically generate password
  - Shareable
  - Stealable

How to fix these problems?

Define: A hash function $H$ is pre-image resistant if, given $y = H(x)$, it is extremely difficult for attackers to find $x'$, s.t. $H(x') = y$.

E.g. SHA-1, SHA-256

The SHA (Secure Hash Algorithm) family is a set of related cryptographic hash functions. The most commonly used function in the family, SHA-1, is employed in a large variety of popular security applications and protocols, including TLS, SSL, PGP, SSH, S/MIME, and IPSec. SHA-1 is considered to be the successor to MD5, an earlier, widely-used hash function. The SHA algorithms were designed by the National Security Agency (NSA) and published as a US government standard.

The first member of the family, published in 1993, is officially called SHA; however, it is often called SHA-0 to avoid confusion with its successors. Two years later, SHA-1, the first successor to SHA, was published. Four more variants have since been issued with increased
output ranges and a slightly different design: SHA-224, SHA-256, SHA-384, and SHA-512 — sometimes collectively referred to as SHA-2.

**Example 1:**

![Diagram of Example 1]

Solved the problem “Password stored in clear on server”.

**Example 2:**

![Diagram of Example 2]

Note: In this case, the sender doesn't reveal foobar but attacker can see $H_1(foobar)$, which is enough to login.

**Data on the wire:**

**Encryption:** Encrypt & MAC all data sent between the terminal and the server.

**Challenge-Response protocol:**
Please start protocol

Attacker

Server

<table>
<thead>
<tr>
<th>Name</th>
<th>Passwd</th>
</tr>
</thead>
<tbody>
<tr>
<td>rob</td>
<td>foobar</td>
</tr>
</tbody>
</table>

Terminal

Rob

foobar

C

r = F(foobar, C)

Security property of F:
Suppose $p$ is password, given
- $F(p, C_1)$
- $F(p, C_2)$
- ...
- $F(p, C_n)$
For $C_1, C_2, ..., C_n$ of the attacker’s choosing, it is extremely difficult for the attacker to produce pair $(C, r)$, s.t. $r = F(p, C)$ and $C \neq C_1, C_2, ..., C_n$.

Server:
1. Set $t$ = current time.
2. Send $C$.
3. if response arrives within 3 seconds, success.

Attacker:
1. Pass $C$ to terminal.
2. Grab response and stop it.
3. Let user try again, then success.
   --- later ---
4. Present original response to server.

*Note: Challenge will not expire.*

Digital Signatures:
- User possesses a private key $K_{priv}$, only the user knows $K_{priv}$.
- World can know a corresponding public key, $K_{pub}$.
- Extremely difficult to compute $K_{priv}$ from $K_{pub}$.
- A signature scheme is two functions $Sig, Ver$, s.t.
  $Ver(K_{pub}, M, Sig(K_{priv}, M)) = valid$
  But an attacker cannot construct any value $X$, s.t.
  $Ver(K_{pub}, M, X) = valid$.

Example1:
**Problem:** No binding between initial authentication and subsequent communication.

**How to fix:**
- Sign all subsequent message, too
- Include $C$ for freshness

**Example 2:**

For real password protocols, please see:
- Encrypted Key Exchange (EKE)
- Diffie-Hellman EKE (DH-EKE)
- Secure Remote Password protocol (SRP)
  - Described in RFC2945: [http://rfc.net/rfc2945.html](http://rfc.net/rfc2945.html)