**C: Ensuring Confidentiality**

**Steganography**

**Encryption / Ciphers**

**I: Ensuring Integrity**

**A: Ensuring Authentication**

**Message Authentication Code (MAC)** is a piece of info used to authenticate another message. (Both Integrity + Authentication)

* Work out a shared secret *“HELLO”* 🡪 Add secret to the real msg *“HELLO WORLD”* 🡪 Generate a hash from it *“57P”*  
  🡪 Send real msg w/ the hash *“WORLD”* + *“57P”* 🡪 Recipient adds secret to real msg, hashes & compares w/ *“57P”*
* You can add another layer of security by encrypting the message, where the shared secret would be the decryption key.

**Keyed-Hash Message Auth Code (HMAC)** is a MAC +

**Public Key Infrastructure**

**Certificate Authorities**

**Digital Signatures**

**Protection against certain attacks**

**Salts** are a non-secret, random value generated to protect against pre-computation attack such as Dictionary / Rainbow attacks

* Concatenated with a password, then hashed together. Hashed output + original salt is stored in the database.
* Even if an attacker knows all the salts and hash function used for a database, they need to check against every salt for each password to find the matching hash value to figure out the password.

**Nonces** are also non-secret, random value generated to be used only ONCE and protects against Replay attacks.

* Ensures that old communications can’t be re-used. Used in authentication protocols
* E.g. eCommerce: attacker could steal information from a victim and use it to make continuous orders. However, if the company receives any orders from the same person with the same nonce, it will discard those as invalid orders.

**Session ID / Session Tokens** is a piece of data used to identify a session (series of message exchanges). Prevents Replay attacks.

* They expire after a short period of pre-set time of inactivity or become invalid after a goal is met.
* Usually in the form of a hash generated by a hash function sent from a server to a client to identify a current session.
* **One-Time Passwords** are similar to session tokens, where the password expires after one use or after a very short time.
* **Time-stamping** is a method to identify when a certain event occurred. Prevents Replay attacks.

**Message Authentication Code (MAC)** is a piece of info used to authenticate another message. (Integrity + Authentication)

* Work out a shared secret *“HELLO”* 🡪 Add secret to the real msg *“HELLO WORLD”* 🡪 Generate a hash from it *“57P”*  
  🡪 Send msg w/ the hash *“WORLD”* + *“57P”* 🡪 Recipient adds secret to msg, hashes & compares w/ *“57P”*
* You can add another layer of security by encrypting the message, where the shared secret would be the decryption key.

**Keyed-Hash Message Auth Code (HMAC)** is an improved version of the MAC, except using a Cryptographic Hash Function.

* Prevents Length Extension Attacks.

**Breaking physical security systems**

**Insider Attack**

**Passive Recon**

* Listen to employee chatter by sitting in nearby café’s or their environment to learn their “language”

**Active Recon**

* Do as much research as possible on the building you are looking to break into.
  + Know all the building plans, security systems such as alarms, the types of locks used, cameras etc.
  + Know the incidence response plans for staff, police or other parties which can stop the attack.
* Know all employees by name, their families, where they live etc. to use it against them.
  + Know the rosters for staff, break times to determine windows of opportunity.

**Misdirection**

* Draw attention of people to a different area than your target area, before launching the attack.
* Disguise yourself as someone else who is supposed to be around the building / area you are targeting.

**Breaking digital security systems**

**Dictionary Attacks** is a technique for defeating a cipher based on trying all strings in a pre-arranged listing, typically derived from a list of words such as in a dictionary.

* Dictionary Attacks succeed because people choose short passwords that are ordinary words / variants.

**Rainbow Table Attacks** is a pre-computed table cracking password hashes.

* Space/Time Trade-off: uses less computing power, but takes up more storage to perform attack.
* How does it work?
  + Databases usually store a cryptographic hash of a user’s password 🡪 no-one can determine the plaintext pw.
  + When users login to the system and put in their password, the pw is hashed and compared to the stored hash.
  + Rainbow Table Attacks store a pre-computed table of hashes,

**Birthday Attacks** is a class of brute-force attack, relying on the Birthday Problem. Generally focuses on finding collisions in HF’s.

* How does it work?
  + Continuously test different inputs with the hash function until you receive the same output h(x) = h(x’)
  + Has broken MD5 hash, Digital signatures

**Replay Attack** A network attack in which a valid data transmission is maliciously or fraudulently repeated or replayed to one or more parties who may process the data as legitimate. (An extended version of a MiTM attack)

* Attackers steals a copy of info from communication between two parties, then replays / reuses it later in a different context for malicious use such as duplicate transactions, circumventing authentication etc.

**Man-in-the-Middle** is a general term for attacks where the hacker is capturing traffic / info between two communicating parties.

* Attackers may try to eavesdrop / intercept data, tamper with messages, block messages etc.

**Length Extension Attack** is an attack where

**Denial of Service Attacks (DoS)**

The most effective DOS attacks are those that have a massive amplification factor.

* Attacks with amplification result in an attacker turning a small amount of bandwidth coming from a small number of machines into a massive traffic load / overwhelming number of packets hitting a victim from all over the world.

**Buffer Overflow**: Crashing an application from overflowing its memory / causing a segmentation fault

* Ping of Death: Sending a ping request that is larger than max size that IP allows, causing an overflow

**Smurf Attack:** Flood a target’s network by sending PING requests to a broadcast address, which broadcasts all messages to hosts connected to the subnet, then spoofing the return address as the victim’s address so they get flooded with replies from the hosts.

**SYN Attack**: An attacker sends a succession of SYN packets / requests to a target system in an attempt to fill up the server’s connection queues and thus denying the service to legitimate TCP users.

1. Attacker sends several SYN packets to the server.

2. The server sends a SYN-ACK packet to the attacker

3. The attacker ignores the SYN-ACK packet, leaving half-opened connections and consuming server resources.

4. Legitimate users try to connect to the server and send a SYN packet, but the server refuses to open a connection with them resulting in a Denial of Service.

**Zero Knowledge Protocol**

**Zero Knowledge Protocol** is a method where one party (*prover*) can prove to another party (*verifier*) that a something is true, without needing to show any information.

* For proof, the protocol must require the verifier to present a challenge to the prover, who needs to respond to the challenge in a way which will convince the verifier that the statement is true.

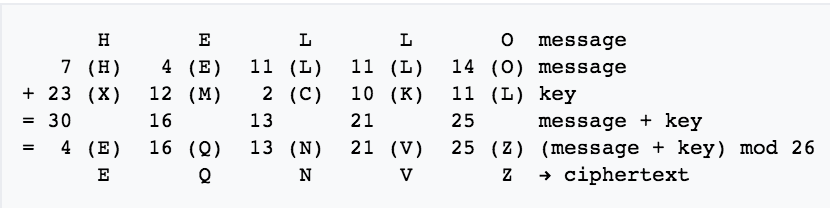
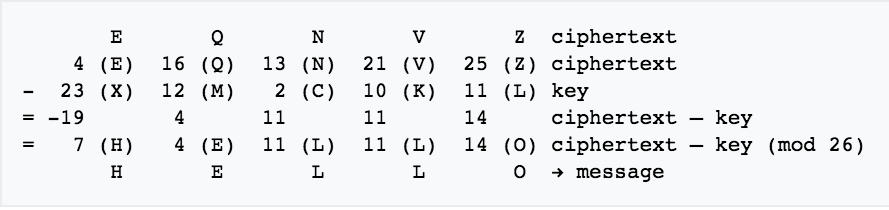
It must satisfy three properties:

* **Completeness**: If the statement is true, the verifier will be convinced of this fact by a prover
* **Soundness**: If the statement is false, no cheating prover can convince the verifier that it is true, except with small %
* **Zero-Knowledge**: If the statement is true, no cheating verifier learns anything other than that the statement made by the prover is true. Just knowing the statement (not the actual secret) is sufficient enough to prove the statement.

**One-Time Pad (OTP)**

**One-Time Pad** is an encryption technique where each character in the plaintext is combined with a character / number from a chosen key on the pad.

* It has a **Perfect Secrecy** property = the encrypted string provides absolutely no info about the original message. (difficult to cryptanalyse / the probability distribution of possible plaintext is independent of the ciphertext)
* Example: We want to encrypt the plaintext “H E L L O”
* Two pads of paper containing identical keys are issued to both parties.  
  ^There is usually a rule to this: e.g. “use the 10th page on 1st Jan”
* The sequence of letters or numbers on the page will be the key for the message.
* Starting from the first letter / number, combine the key and message using **modular addition** (mod 26)
* Decoding is just using **modular subtraction**.

**Revision Lecture Notes**

Possible questions

* Whistle blowing question 🡪 Talked about whistle blowing, consequences of whistle blowing
* Hutu incident 🡪 Two tribes in African having a falling out 🡪 Lead to the Rwandan Genocide
* One-time pads
* Zero knowledge protocol

Difference between a hash function and a cryptographic hash function

* A cryptographic hash function is more special because it is a hash function which has certain properties which make it suitable for use in cryptography:  
  **Pre-image resistant, Second Pre-image resistance, Collision Resistant**
* A cryptographic hash function must withstand all KNOWN types of cryptanalytic attacks

Difference between hash function and encryption function

* Using a Hash Function is like baking a loaf of break.
  + Extremely difficult to reverse. You are only using the output hash as a comparison tool.
  + Many:1 one-way mapping
* Using an Encryption Function is like having a safe deposit box.
  + What you put in there needs to be taken back out, as long as you have the key to unlock the box.
  + Symmetric. 1:1 mapping

MD5: A widely used algorithm / hash function producing a 128bit hash value. (but with 56bits of security which is too small)

* Problems:
  + Hash size is too small, so anything protected with that hash is subject to to attacks on the hash.
  + E.g. Collision attacks can find collisions within seconds using a Pentium 4 processor (224.1 power)
  + #bits shaved off because of some weaknesses 🡪 #bits is not good enough 🡪 Easy to do a Birthday Attack on it to break it
* Not-collision resistant + Not suitable for SSL Certificates / Digital Signatures
* Say you’re given an MD5 hash (128bits) and someone asks you to find the thing that hashed this, you got no chance
  + Using an MD5 hash for your passwords is actually okay, its not bad to use it, because it is still hard to extract your password.
  + Birthday attack was used against some Certificate Authority who were using MD5 hashes for their SSL Certificates and a research company was able to generate a fake SSL Certificate because they found that they were able to do a birthday attack

SHA1: Hash Function, for securing integrity (not confidentiality)

* Problem: Too small, already been broken. Collision has been found.

SHA2: Works same way as SHA1 but is stronger / generates a longer key

* Problem: Small, but hasn’t been broken yet.

SHA3 (256 / 512bits):

* Good because it goes up to very large sizes
* No known attacks on it yet

**Output key-size doesn’t always = bits of security**

* **There are significant differences in level of security of hash functions against cryptanalytic attacks**.
* **Length of a hash is usually a problem, but there is more to it.**

When you salt a password, where do you store the salt? 🡪 You store the salt with the hash.

* Hash the message + store the hash and the salt next to each other.
* The salt would just be in plainview, it is not secret because its just used to stop people from attacking multiple things in parallel i.e. a Dictionary Attack / Rainbow Table Attack