**WEEK 2**

**Active Recon**: Putting a bug somewhere to listen. More quick + effective but more detectable.

**Passive Recon**: When you don’t do anything and passively get the info (Eavesdropping etc.)

**Asymmetry of Attack and Defence:** Easy to attack and hard to defend

**M&M’s**

* Manage complexity by NOT THINKING ABOUT IT
* Create pockets of safety. You cannot always trust people inside your organisation

**Kerckhoff’s Principle**

* A cryptosystem should be secure even if everything about the system except the key, is public knowledge.
* Assume that the attacker knows the system

*Null Hypothesis is when there is no statistical significance between two variables*

**Type 1 Errors**: When the Null Hypothesis is TRUE and you reject it.

* Example: Medical researcher rejects null hypothesis and concludes that two medicines are different. In fact, they are the same and this error won’t matter much because patients will benefit equally from both.

**Type 2 Errors**: When the Null Hypothesis is FALSE and you fail to reject it / accept it.

* Example: Medical researcher accepts null hypothesis and concludes that two medicines are the same. In fact, they are different and this error may be life-threatening if the less-effective medication is sold to the public than the more effective one.

**Simple Substitution Ciphers**: Operates on single letters

* Caesar Cipher is a shifting cipher. E.g. ROT13 (13 rotations)

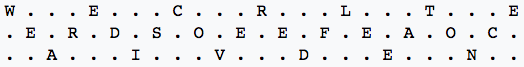
**Polygraphic Substitution Cipher**: Operates on larger groups of letters. Uses substitutions at DIFFERENT POSITIONS in the msg.

**Monoalphabetic Substitution Cipher**: Uses fixed substitution over the ENTIRE message (Caesar is also monoalphabetic)

CIA Protocols

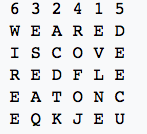
* **Confidentiality**: Anyone can see or feel it, but only the intended recipient with the key can get the message
* **Integrity**: How can we check that a message hasn’t been tampered with? I.e. Data Integrity
* **Authentication**: How do we really know the message came from the real person, not a spy/fake?

How to ensure confidentiality

* **Steganography**: Concealing messages inside other things. E.g. tattoo message on slave’s head.
* **Substitution**: Change the secret / key. Broken through **Frequency Analysis**  
  NOTE: **E T A O I N SHRDLU** = 12 most frequent letters in English text.
* **Transposition**: Change letters and move it around  
  A cipher where positions of chars are moved around so that it becomes a permutation of the plaintext.  
   **Rail Fence Cipher**: Plaintext is written diagonally up and down  
   

**Route Cipher**: Similar to Rail Fence, but may “spiral inwards, clockwise, etc.”

**Columnar Transposition**: Message is written out in rows of fixed length (determined by a written key e.g. ZEBRA = 6 letters per row) where permutation is defined by the alphabetical order of letters in the keyword.



**Playfair Cipher** is the first diagram substitution cipher (encrypting pairs of letters).

**Converting #bits to factorial**: E.g. 170 operations = 5 \* 4 \* 3 \* 2 \* 1 = 120 = requires 7bits to brute force.

**WEEK 3**

Public Keys (**Asymmetric Key Cryptography**)

* One key for writing messages
* One key for reading messages
* Everyone keeps their reading key secret, but they all share their writing key that corresponds to their reading key.
* E.g. Anyone in the world can talk to you, but only you can read the messages.

**gx % m = p shared secret: P private key: x**

**1. Alice and Bob pick a PRIVATE KEY X**

**2. Given the same generator (g) and primitive mod (m), they calculate their remainder using X**

**3. Using their remainders combined with their PRIVATE KEY (x), they generate a SHARED SECRET (P)  
4. The SHARED SECRET would be used to send messages to each other.  
5. Even if the spy can see the shared secret, it would be difficult to find the PRIVATE KEY X**.

**Problems with the Diffie-Hellman method**

It is subject to **Man-in-the-Middle (MitM) attacks**.

* Alice and Bob don’t know that a Spy is watching their communications!
* PART 1:
  + Spy intercepts Bob’s shared secret **Bp** with Alice.
  + Spy sends their own shared secret **Sp** to Alice.
* PART 2:
  + Spy intercepts Alice’s shared secret **Ap** with Bob.
  + Spy sends their own shared secret to **Sp** Bob.
* PART 3:
  + Any message sent out by Alice is decrypted using the **fake shared secret Sp**
  + The message is either read or modified before re-encrypting + sending to Bob
  + Same goes for Bob sending out messages to Alice
* **This vulnerability is present because the Diffie-Hellman key exchange does not authenticate the participants.**

**WEEK 4**

**Moral Hazard**: When someone has incentive to act the wrong way even though they’re supposed to act the correct way

**Bits of work**

* If a key is 128 bits long, there are 2128 different keys.
* On average, you have to search half of these keys 2127 before you get lucky.

**Salt in Cryptography**

* A salt is random data that is used as an additional inputs to a one-way function that “hashes” a password.
* The primary function of salts is to defend against dictionary attacks.

**Dictionary Attacks**

* 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.
* It only tries possibilities which are deemed most likely to succeed.
* Dictionary Attacks often succeed because many people have a tendency to choose short passwords that are ordinary words / common passwords / variants.

**Message Authentication Code (MAC)**

* A short piece of information used to authenticate a message.
* The **MAC value protects both a message’s data integrity as well as its authenticity** by allowing verifiers (who also possess the secret key) to detect any changes to the message content.

**Non-repudiation**: You can’t claim that something didn’t happen when it did happen. E.g. signing contracts.

**Replay Attacks**: 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.

**Nonce**: A NUMBER USED ONCE.

**Length Extension Attack**: Many cryptographic hashes are iterative (take one part of the message, do something, then take the next part of the message, do something so on).

* If you know the hash of the whole phrase, you can just add something to the end of the hash  
  (since the unknown is only at the beginning)
* **HMAC** puts the hash of the first half at the end, then hash everything again.

**Hashing**: How it works

* Sender and receiver agree on a codeword / secret string
* Sender appends secret to plain text + passes plaintext into a hashing algorithm = generate a hash
* Sender removes the secret from the plain text and sends it to receiver along with the generated hash
* Receiver appends secret to the received plaintext and passes plaintext into the same hashing algorithm
* The resulting hash is compared with the received hash to **authenticate the sender** and ensure **integrity** **of the message**

**Cryptographic Hashes** should have chance of collision equal to 2n where n = size of the hash in bits.

* E.g. 256bit hash = 2256 chance of collision.
* It has all the properties of a normal hash + can’t easily be reversed to obtain the message.