**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.
* Always assume that an 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.

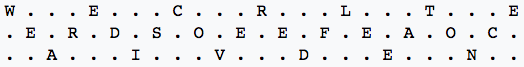
**Simple Substitution Ciphers**: Operates on single letters

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

**C.I.A Protocols**

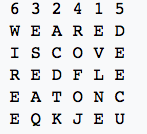
* **Confidentiality**: Anyone can see or feel it, but only the intended recipient with the key can get the message
* **Integrity**: How can we check that a message hasn’t been tampered with? I.e. Data Integrity
* **Authentication**: How do we really know the message came from the real person, not a spy/fake?

**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 input to a one-way function that “hashes” a password.
* The primary function of salts is to defend against dictionary attacks.
* Take advantage of **space/time trade-off** and can only attack 1 person but not everyone.
* Attacking everyone means they have to create one dictionary for each person, since everyone’s salt value is different.

**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.

**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.

**WEEK 5**

**Side-Channel Attack** is an attack based of information gained from the physical implementation of a cryptosystem, rather than from brute force or theoretical weaknesses in an algorithm.

* *Timing Attack, Power Consumption, Electromagnetic Leaks, Sound* can all sources of extra info which can be exploited.
* E.g. Voting for Trump vs. Hilary
  + You have to write either Trump or Hilary on a piece of paper.
  + Side-Channel Attack = watch someone write Hilary’s three-word-name vs. Trump’s easier two-world name  
    ^Timing Attack. You can see who they voted for.

**Advanced Encryption Standard (AES)** is a good algorithm, but can be cracked by side-channel attacks.

**Bugs, Vulnerabilities and Exploits**

**Vulnerability** is a bug in a system which would potentially let someone attack the system in a way that you don’t intend it to behave.

**Memory Map** is a structure of data that indicates how memory is laid out in a computer.

* Memory can be 4 or 8 bytes wide, depending on a 32bit / 64bit computer.
* Programs are usually stored up HIGH in the memory map (towards 0x000000 address)
* Programs usually store variables in the LOWER areas in the memory map (0xFFFFFF)

**Stack Frame** is an area of memory stored temporarily when a program carries out a function.

* Frames are stored in the STACK = the lower / bigger memory address (towards 0xFFFFFF)
* As we create variables, the stack frame becomes bigger / extended area of memory while the function is running

**Return Address** is an address in memory stored inside a function’s stack frame.

* It tells a function where to go next after the function finishes executing + start executing the program that the return address points to.

**Stack Overflow** is when you override the return address with your own data, by accessing memory beyond the stack’s bounds.

* You can **exploit** this vulnerability by placing a bad program in a location in memory, executing the overflow and directing the return address to your bad program.

**NOP-Sled** **(NOP = No Operation, an instruction that does nothing)**

* Purpose is to slide the CPU’s instruction execution flow to its final, desired destination.
* It helps circumvent **stack randomisation** in a buffer overflow attack
  + Stack randomisation attempts to randomise where items are in memory, to make injecting malicious code more difficult as some buffer / stack overflow attacks depend on knowing where items are in memory.
  + An attacker can place a NOP-Sled in a large range of memory. If the program jumps anywhere into the sled, it will run through all the NOP’s and do nothing until it reaches the **payload code**.
* A NOP-Sled basically makes the target address bigger, so that the code can jump anywhere in the sled and go towards the payload.

**Heap** is a memory location in a computer set aside for dynamic allocation. E.g. when you MALLOC

* It is stored roughly in the MIDDLE area of memory.
* Adding things to the HEAP = goes downwards towards 0xFFFFFF
* Adding things to the STACK = goes upwards towards 0x000000
* Eventually, the Heap / Stack may collide and you will get an out of memory error.
* To overflow the Heap, you can overflow critical data below the Heap instead of the return address in the Stack Frame

**Printf Vulnerabilities**

* Printf vulns are bad because it allows an attacker to write to memory through printf.

**Shellcode** is the actual code that you want the exploit to run that achieves the effect you want. (It is typically the payload)

**Common Vulnerabilities and Exposures (CVE)**

* A database of vulnerabilities with CVE ID numbers attached to each vulnerability.
* As vulnerabilities get found and reported, they are given ID no’s and certain companies / naming authorities such as Google, Microsoft can hand out these numbers, which are all put into a centralised database.

**Responsible Disclosure:** Disclose vulns to the software vendor and give them a period of time to do something about it.

* If software vendor doesn’t listen, report to media or a CERT (Computer Emergency Response Team) or to a responsible person / company to try get a CVE number.

**Assets**

Strategies for identifying assets:

* **Get lots of eyes looking at the asset, including people who own the assets. See what they all value about it.**
* **Develop a sensible plan – well designed to tease information out of them** (humans are good critics)
* **Periodically revise current list of assets**

**Tangible Assets**: Gold, murder weapon (physical evidence), car.

**Intangible Assets**: Employee morale, customer info, company secrets, availability of services

Strategies for assigning values to assets:

* Survey what people think. No single person should be solely evaluating assets.
* “How much money would you lose If this data centre was to go down for 24 hours?”
* “How much will you lose if your company is disconnected to the internet for 3 hours?”

**Bits of Security**

**Entropy**: A measure of chaos. English has low Entropy because on a lot of patterns / ordering.

**Bits of Security** is a measure of work to break some security.

* 10 bits of work / security = 210 operations to break the security.
* Example QN#1
  + How many bits of work / security would it take to break a key of length 7, comprised of alphabetical letters
  + 267 combinations to brute force the key (worst case). Convert power of 2: **267 🡪 327 🡪25^7 🡪 235**
  + **35 bits of work to brute force the key**. (This is crackable. Anything > 128bits make it too difficult for attackers)
* Example QN#2
  + How long would it take to crack 60 bits of work? What if attacker’s speed is 250 operations per second?
  + **60 bits of work = 260** / Attacker needs to spend 260 – 250 = 210 seconds to crack it

**Hash Functions / What makes a good Hash Function?**

3 properties that a cryptographic hash function should have:

1. **Pre-image resistance**
   * Given an output y, it is computationally infeasible to find an x such that h(x) = y
   * Given the hash, you can’t reverse it to find the input.
2. **Second pre-image resistance**
   * Given an input x, it is computationally infeasible to find any SECOND INPUT x’ such that h(x) = h(x’)
3. **Collision resistance**
   * Not given any input, it is computationally infeasible to find any two distinct inputs x / x’ that hash to the same output.

**High-Resistance Hash Function:** If your attacker has to use **Brute-Force** to solve the problem.

**Low-Resistance Hash Function:** If your attackers shows any way to do it faster than Brute-Force.

**Pre-image attack** is an attack on a one-way function. A function which when given the hash y, you can find the input x.

**Birthday attack** is an attack which finds collisions in one-way functions.

* Based off the Birthday Problem and the Pigeonhole Principle.

**Random Mapping**: A small change in the input (1bit) should ideally change on average half of the output.

**WEEK 8**

**Bits of work walkthrough**

Cracking passwords example:  
  
Guess the 6 char password of someone – where the password is all in lower case, except possibly the first letter can be either upper or lowercase.

1. Think about the main variables in the question  
   **[A-Z + a-z][a-z][a-z][a-z][a-z][a-z]**
2. Make the appropriate calculations to find a approx. number.  
   **52 x 26 x 26 x 26 x 26 x 26 = ~617 million**
3. Go to the nearest number which is a power of 2 + convert to power of 2  
   **617 = ~512** **(closest power of 2 value)  
   1 million = 220**  
   **512 = 29**  
   **Therefore, 220 + 29 = 229** **bits of work**
4. To find the average case bits of work, divide by 2  
   **229 bits of work / 2 = 228 bits of work**

Bits of work – should we consider Average or Worse case?

* **ATTACKER POV = Worst Case**
* **DEFENDER POV = Average Case**

**Identity and Authentication**

Authentication is the hardest part of the C.I.A security properties

**Authentication**

* You are in a prison, where there is a computer screen is the only interaction you have with the outside world.
* If a message appeared, saying “It is mum, tell me the password to get into our house”.
* How do you know it is really her / how do you authenticate the person?
* **A shared secret is the main key to distinguish between an authorised person and an attacker.**

How do we authenticate the person?

* Add randomness to the shared secret to make replay attacks more difficult
  + **Replay Attacks**: An attack where transmitted data is copied and repeated to one or more parties, causing the recipients to process the replay data as legitimate.  
    E.g. attacker replaying an expired response to gain privileges
  + One countermeasure = adding a **Nonce (number only used once)**, which is random / pseudorandom
* Replying with a **Challenge Response Authentication**
  + You need to authenticate someone who says they are your mum asking for your credit card details to borrow money.
  + You ask her a personal question that only you and her would know as a challenge-response.
* **Multi-factor authentication**
  + **One-Time Password SMS**
    - A single-use password that is sent to you via. SMS as part of the authentication process
  + **2-Factor-Authentication**
    - A two-step authentication process that requires not only a user/password, but also something that only the user has on them.
    - E.g. mobile phone OTP SMS for online banking, or physical token to gain access to a venue maybe

**3 Ways to Authenticate**

**1. Something you ARE**

* This property is part of you as a person
* Biometrics
  + Fingerprints or Iris (eye scanning)
* Limitations
  + It is unchangeable. If someone gets a copy of your fingerprints, you cannot change your fingerprint to a new one.

**2. Something you HAVE**

* This property is something that you have as a good
  + Driver license, password, key card
* Limitations
  + These are forgeable (Fake ID / password)
  + It must be with you when you use it.

**3. Something you KNOW**

* This property is something that only you would know, but no one else
  + Password
* Limitations
  + It can be used by someone else
  + You can forget it
  + It can be discovered by brute force or packet sniffing / eavesdropping
* Protection against limitations
  + Add a password salt
  + Improving protocol to prevent sniffing / eavesdropping i.e. encrypting all network traffic.

**Simple Authentication Protocol Examples**

* **S/KEY** is a One-Time-Password system
  + Begins with a secret **W**
  + A cryptographic hash function **H** **is applied *n* times to W**, producing a **hash chain of *n* one-time passwords**
    - Result would be H(W) 🡪 H(H(W)) … 🡪 Hn(W)
  + The initial secret W is discarded
  + The **user keeps the list of *n* passwords**, while the **server only keeps the final hash Hn(W)**
  + To authenticate, the user provides the server with Hn-1(W)
    - The server computes Hn-1(W) and compares with its Hn(W) value.
    - If Hn-1(W) produces Hn(W), then authentication is successful.
  + Security of S/KEY relies on the **difficulty of reversing cryptographic hash functions**.
* **SKID**
  + Hk = a keyed MAC. Both A+B know k.
  + A chooses random no. Ra and sends to B
    - **A ----------- Ra ------------> B**
  + B chooses random no. Rb and sends Rb, Hk(Ra, Rb, Bob’s name)
    - **A <---------- Rb, Hk(Ra, Rb, “BOB”) -------- B**Ra is needed to prevent a replay attack  
      Rb is needed to prevent a MiTM attack
  + A sends Hk(Rb, Alice’s name) to B
    - **A ------------- Hk(Rb, “ALICE”) --------> B**
  + Now each party knows their shared secrets
* **Interlock protocol (RSa)**
  + A encrypts her message with B’s key + sends half of the encrypted message to B.
  + B encrypts his message with A’s key + sends half of the encrypted message to A.
  + A sends 2nd half of her message to B // B sends 2nd half of his message to A
  + *Strength of this protocol lies in the fact that half of an encrypted message cannot be decrypted. Attacker needs both halves to read it.*

**Authentication vs. Authorisation**

* **Authentication** = who is this person?
* **Authorisation** = Does this person have permission to do something?
* Airport Example:
  + Authentication = Check-in + boarding pass
  + Authorisation = Check if the passenger goes to the correct flight

**TOCTTOU (Time of check to time of use) Error**

* A software bug caused by changes in a system between the **checking of a condition** (e.g. security detail) and the **use of the results of the check**.
* Example: Wikipedia, where users are allowed to edit pages + admins can also lock pages from being edited.
  + A user decides to edit the page, opening up a form which allows them to alter content.
  + Before the user submits the edits, the Admin decides to lock the page.
  + Since the edit form has already begun, when the user submits the form, it will accept the edits.
* **User begins edits (authorisation CHECKED) 🡪 Authorisation USED later, when edit shouldn’t be allowed**
* TOCTTOU races can be used for privilege escalation.
* Implications
  + Applications cannot assume that the state managed by the operating system will not change between system calls.

**Identity Theft**

Identity theft is the misuse of personal information to do things on a victim’s behalf without their consent.

Reasons why people don’t report misuse of personal information

* Embarrassment
* Unaware they have been a victim of a crime
* Confusion as to the agency to which they should be reporting the incident to
* Did not lose money and therefore do not believe they need to report it
* Do not believe police will be able to do anything about it

**What do they do with your data?** (Most popular target = ID Cards / Driver’s License)

* Loans
* Social Security
* Crime
* Tax
* Driving
* Buy phones
* Credit cards
* Social damage

**How to respond to identity theft?**

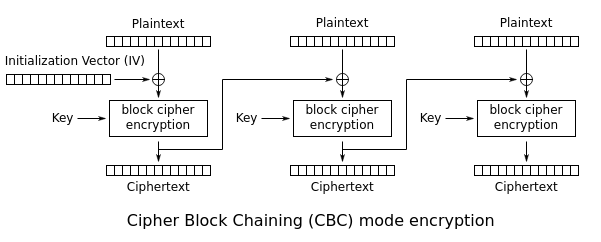
* Report crime to police 🡪 Contact IDCare victim support 🡪 Contact credential issuing org 🡪 Alert your bank  
  🡪 Get a copy of your credit report 🡪 Close all fraud accounts 🡪 Consider legal assistance 🡪 Getting a Victim’s Cert  
  🡪 Report privacy breach to Aus Privacy Commissioner

**Privacy**

The main problem with privacy is how to have a balance between privacy and security.

**Chain Block Cipher (CBC)**

Split plain text into multiple blocks, then use ciphertext of the previous block XOR with the current plaintext (for the first block XOR with Initialization Vector (IV) instead).



If the first block has index 1:  
  
The mathematical formula for **CBC encryption** is:

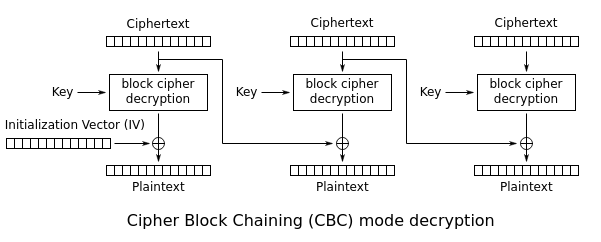
https://lh4.googleusercontent.com/VXIb-d0DRRCfyFLZx25Xn9wBZ_Tjg2iAGfjrsHgaW6pBlOxvh0RRc-OLJ_MLTiB3njkJil_kUCmWWF0Dyh9K1AnpgQYycwjlJQJL6dyrOT_FixTZdz-N7PerXPLtqmPSF7V-K-XE

https://lh4.googleusercontent.com/qieIWrmFKa_PuSmKeMsLKBblekM5WnFdBM9qnqzkaOSjHhrR3pnd3_XeIYEuwnKkfMKl2pjNPhlgG6Km8pHSy4Fwy_rIPxsxEsC8fv6u3F3KgIIhkgzU1040M1lPEWD5mrNFLap-

The mathematical formula for **CBC decryption** is:

https://lh6.googleusercontent.com/5AR3yp_MKA1Dziiu1EPSAJhd2gz0h0S4o0eBEF59mwqxzS3xlBlvX85SnDe_MEi2Wzk_KvyNY3AxVFWqsi-tOfCPfLHRb3plssUBz7wCUP-L-XvdZXjMZ8quRIK0hsTpfKx0mk_c

https://lh3.googleusercontent.com/IaJd_eorvq68yvem5zl9FRw7nSTiV0GlLINTRJuT7FwzIyxoh5u1NM3p6ohE3D5d60xYeH2P1TtiCteaxv5xql82q2QyQfqZ_4NyYJIZyBguzc2SbwIpjoIKMYiyfK9zLIIilj-C



**WEEK 9**

**Data** and **control** are often combined

* E.g. self driving car that enforces speed limits.
* As soon as you have a system that makes control decisions based on data, you have an attackable system.

**Trust and problem with key distribution**

How do we send keys around? How do we know what we are authenticating?

Two approaches to authentication:

1. **Centralised Authority based Command and Control (e.g. PKI)**
   * Weakness: Single-point of failure
2. **Decentralised peer based web of trust (e.g. PGP, Blockchain)**
   * How do I know Richard is Richard? I know because the university admins know and I trust the uni admins.

**Public Key Infrastructure (PKI)**

PKI is a set of *roles, policies and procedures* needed to *create, manage, use, store and revoke* **digital certificates** and **manage public-key encryption**.

* **Purpose =** facilitate secure electronic transfer of data for a range of network activities
  + Network activities such as eCommerce, internet banking, confidential email.
  + Required for activities where simple passwords are inadequate for authentication + **more rigorous proof is required to confirm the identity of the parties** involved in the communication AND to **validate the information being transferred.**
* PKI is generally used on the internet (SSL Certificates)
  + SSL is one of the many transport mechanisms that allows PKI to work over the network.
* What’s the problem with asking for a person’s public key and then using it to encrypt?
  + MiTM attacks. Even could intercept messages from Alice 🡪 Bob and send their own public key to Bob and pretend to be each person.
* **Solution = a “bank” of public keys = SSL Certificates**
  + **Certificate Authorities (CA)**
  + Web browsers come preloaded with the public keys of the certificate authorities.
  + Bob sends Alice a certificate that has been signed using the CA’s private key.  
    Using the CA’s public key, you can figure out that the certificate is valid.
* **Certificate Authorities**
  + A CA certifies the identity of an individual through extensive vetting and if satisfied, provides a signed certificate confirming the identity, encrypted with a private key held by the CA.
  + Anyone that communicates with this individual can use the CA’s public key to decrypt it and verify the certificate.
* **Bruce Schneier: 10 risks with PKI’s**
  + Why do we trust + why do we trust them?
  + Who is using my key?
  + How secure is the verifying computer?
  + Which Richard Branson is he?
  + Is the CA and authority?
  + Is the user part of the security design?
  + How did the CA identify the certificate holder?
  + How secure are the certificate practices?
  + Why are we using the CA process anyway?
* With web browsing, we are often only authenticating a URL.
  + There is every chance the person or company we want to communicate with is NOT the URL we are authenticating.
  + E.g. [www.google.com](http://www.google.com) is different from [www.go0gle.com](http://www.go0gle.com)
* **SSL Certificates**
  + Typically, an SSL Certificate will contain:
  + Your domain name, company name, address/city/state/country, expiration date of your certificate  
    + details of the Certificate Authority responsible for issuing the certificate.
* **Transport Layer Security (TLS)**
  + TLS is a protocol that ensures privacy between communicating applications and their users on the internet.
  + When a server and client communicates, TLS ensures that no third party may EAVESDROP or TAMPER with any message.
  + TLS is the successor to SSL.
* **HTTPS**
  + HTTPS = Use of SSL or TSL as a sub-layer under regular HTTP application layering.
  + Encrypts and Decrypts user page requests as well as the pages that returned by the Web Server.
  + Protects against EAVEDROPPING and MiTM attacks.

**Perfect Forward Secrecy**

If bad guys are observing communication between two parties and one day, they discover the public key, wouldn’t it be nice to have **an algorithm that ensures the safety of all** **past communications?**

* This is the idea of **Perfect Forward Secrecy**
* PKI is expensive / slow
* How do we solve this problem?
  + You create a public key channel between each other (slow / expensive)
  + You agree on the parameters for Diffie-Hellman
  + You work out what the session key is and then you use that.

**HTTPS** for example

* **Session Key** and **Symmetric Cipher** for Perfect Forward Secracy
* Uses **PKI asymmetric cryptography** and authenticate and setup a communication stream.
* Then the server and client uses this secure infrastructure to generate a symmetric session key, for example **Diffie-Hellman**, that is tossed at the end of each session.
* So a new key is used for every session.
* **MAC** for **integrity**.

**Cache Poisoning**

**Address Resolution Protocol (ARP)** works out which devices (by MAC address) are sitting on which IP addresses on your local network, stored in ARP tables.

* When a message comes in for a particular IP, it knows which MAC address it is going to.
* If you can attack the switch on the network, send a message to your own machine rather than the designated MAC address.
* Steal all the messages from your neighbour and send it back to yourself.
* ARP tables are stored on the switches for the network, but these caches can be “poisoned” with fake entries.

**Work out what are the main security focus / issues with the case before trying to solve (WHICH ONE IS IT: C . I . A ???)**

**Block Ciphers**

When encrypting large amounts of data, the data is split into blocks and encrypted, then combined together to form the cipher text. Block modes are the ways this can be done.

1. **EBC (Electronic CodeBook)**
   * You have a message that is broken into 3 blocks. You encrypt each block independently and the concatenate them together 🡪 the encrypted cipher text
   * Problems with ECB
     1. If you’ve got a big plaintext with any patterns / order, it is likely that you have blocks with identical content.   
        Identical plaintext block 🡪 identical cipher-text blocks 🡪 Easy to crypt-analyse and reverse engineer
   * The different blocks should be treated differently to prevent this.
2. **CBC (Cipher Block Chain)**
   * The plain text is XOR’d with the previous ciphertext block before being encrypted
   * The first block is encrypted using an **Initialisation Vector (IV)**
   * Problems:
     1. When encrypting, each block can only be encrypted once the previous block has been encrypted (non-parallelised / can’t do all blocks at once hence slower)  
        However, you can decrypt in parallel because you have all the cipher-text before you decrypt.
     2. The message must be a multiple of the cipher block size

It is vulnerable to **Padding Oracle Attacks**

* + 1. When decrypting, if the wrong IV is used on the first block, it becomes corrupted. However, each subsequent block will be correct. This is because it uses the previous cipher text to XOR.

1. **CTR (Counter)**
   * You fold in a **nonce / counter** each time you encrypt.
   * You encrypt the nonce using the key and XOR it with the plaintext. You are actually encrypting the nonce rather than the plaintext
   * However, if you are XOR-ing two things, it will have the same properties as the plaintext
   * Nonce/Counter 🡪 Encrypt with key 🡪 XOR encrypted nonce/counter with the plaintext = CIPHERTEXT

Nonce/Counter 🡪 Encrypt with key 🡪 XOR encrypted nonce/counter with the ciphertext = PLAINTEXT

* + The counter:
    1. Can be any function which produced a sequence that is guaranteed not to repeat for a long time.
    2. Increment-by-one counters are the simplest and most popular.
  + The IV / nonce:
    1. While it is better to be random it does not have to be
    2. In the case of a non-random nonce (e.g. packet counter), the nonce and counter should be concatenated to maintain security.
  + Encryption / decryption speed
    1. Using this method, encryption / decryption is parallel is possible as long as you have the nonce and counter for that block.
  + Problem: if a non-random nonce is used and not concatenated with the counter, it becomes very vulnerable to an attack.

**PGP (Pretty Good Privacy)**

In the PGP system, each user has a publicly known key + a private key only known to the user (similar to RSA)

Since encrypting long messages using RSA can be time consuming, PGP uses a faster symmetric encryption method and encrypts that key using the recipients public RSA key.

It can be used for authentication as follows:

* A message (data that we want to receive) is hashed 🡪 we get a **Message Digest** 🡪 this is encrypted using the sender’s private key to get a **Signature**
* The signature is concatenated with the message and sent to the recipient.
* The recipient reverses the process using the public key of the sender to get the message digest and hashes the message it received to ensure it matches the message digest (Authentication).

The weakness of PGP is that you have to be sure that the public key you’re using belongs to the person you’re trying to authenticate.

**WEEK 10**

**Incidence Response**

* In an exam, we can be asked about a random real-world scenario / event
  + *If I were the Head of Security / Head of IT when WannaCry began to break out, what would I have done?*
* The **Benefit of Hindsight**
  + In retrospect, when you know the consequences, you would always know the right solution to a situation
  + When you’re doing incidence response, you don’t know what the right response would be
* Having a **Measured Response**
  + If you break the fire-alarm glass, would you save lives?
  + VS. Would you just annoy your boss because it costs a lot of money to replace?
  + What is the correct move?
  + *You have to think about a measured action to take and the consequences of that action*.
* Many companies have cut links with old manual systems.
  + This can be bad, because if new systems break, then there would be nothing to work off. Your entire system would be frozen.
  + E.g. in the future with no pen and paper, if computers break down, there would be no way to communicate.

**Time of an attack**

* The best time to attack would be Friday Afternoon, New Years, Christmas.
* *A good attacker will NOT attack when the victim is prepared.*

**Zero-Knowledge Protocol**

* Let’s you know something without letting anyone know that you know it
* Find an example???