**MORNING**

**Bits of Work:**

* Substitution cipher: A -> 26 choices, b -> 25 choices, c -> 24… and … so for the alphabet, there’s 26! different ways of substituting an alphabet
  + 26! = 4 \* 10^(26), huge number -> 26 **d**its of work; 1000 = 10 **b**its (2^10) and 1000 is 10^3. So 10^26 = 10^3 \* …. \* 10^3 [8 times] \* 100 = 87 bits, big work
* We never really care about the # of bits, we just want to show the time is ridiculous if you were to brute force
* But we solve these in minutes with frequency analysis -> entropy is the opposite or patterns, so if we introduced entropy (randomness), brute force might be best way
  + If English had no patterns i.e. “I’m hungry” -> “I’m hungry” but “I’m hungry now” -> my car is red, we would have complete randomness, with no relation between letters/words, we would need to brute force
  + We know that theoretically, a 5-letter word has [26][26][26][26][26] = roughly 32^5 which is 2^25 possible “words” [trillions, but English only has 8800 5-letter words [13 bits of information]. Per letter 13/5 = 2.5 bits of info.
    - For a 100-letter word: 2.5 \*100 really, not the naive 26^100

**Protocol – Telegraph: how to not have people tamper with your messages**

* Say we are withdrawing from the bank. We don’t really need a message to be confidential, it’s okay if I say I want to withdraw, but we don’t want tampering
  + How can we make a tamper evident communication protocol? HASHING
* HASHING: summarise one thing with another thing
  + Hash function has to give the same results every time for the same inputs
  + E.x. 6841 black, 6441 red, P’grad -> heart/spade; U’grad -> club/ diamond
    - Value -> King = 0, Ace = 1, 1-10, Jack = 11, Queen = 12
      * My birthday = 3rd, mod 13 => 3
    - We can just check the hash value not the data to see where things are
    - 2 people had the queen of clubs -> this is hash collision, happens since we had 52 hash values, for 150 students, so we must have clashes
      * Lots -> little info. If the 2 Queens of clubs swapped locations, although they are different people, we don’t detect a change
      * Once we have sample size = sqrt(hash size), we get a 50/50 change of collision **(Birthday Attack).**
  + Solve telegraph problem with a hash -> everyone knows the hash function
    - Do a hash that is un-reversable? (Homework for tonight’s lecture)

**EVENING**

**The Social Engineering Presentation:**

**Intro:**

* Relies on human interactions; common examples: phishing, ransomware, USB baiting (drop malicious usb somewhere likely to be picked up) etc.
* 95% of all attacks involve social engineering
* It takes about 146 days to detect a breach caused by social engineering

**Life Cycle:**

* Investigation - learn as much as possible about the victim
* Hook - initiate the conversation with the target and build rapport
* Play - obtain the information
* Exit - leave the conversation without seeming suspicious

**Social Engineering Vectors:**

* Pretexting - invented scenario to get information out from someone
* Baiting - taking advantage of curiosity or greed
* Quid Pro Quo - getting something for something
* Tailgating e.g. heavy box technique
* Phishing - digital invented scenario e.g. email to people to get their credit card details. Phone phishing is also easier now because we can automate voices

**Security Questions:**

* Humans often struggle to produce secure and unique passwords
* Social media makes it easier to answer security questions
* Pseudo-private information - information is less private for friends and family
* Prevention and strategy:
  + Always lie - consider a security question as another password
  + Use password managers and scrub your social media
  + Don’t reuse security questions and answers

**Principles of Persuasion:**

* People can be easily exploited using persuasion
* Reciprocity - do someone a favour, they feel obligated to return the favour
  + Can’t make it look like a bribe
    - ^ the time delay between the initial gift and the later request
* Liking: If you can get someone to like you, it’s much easier to influence them
  + Presentation, body language, establish rapport etc.
  + E.g. brands on Twitter are relying on humour to drive sales
* Social Queues: People believe in the social queues around them
* Authority: People often blindly follow authority figures
  + Can influence other people based on what you wear & say, how you act etc.
  + E.g. 2007 Chaser’s War - bought expensive -> able to get quite far into a secure area, even though they had fake ids

**Other:**

* Dumpster Diving - can go through rubbish and find lots of information
* Even after you burn confidential documents, you can obtain info from the ashes

**HASHING: Continued:**

* Does it matter if someone knows we are withdrawing, and the bank is approving? Not really, confidentiality is not massively important **(C).**
* Desired resistance against attacks:
  + Preimage resistance: given a hash value y = h(M), it is impossible to find the original M. I.e. cannot go hinverse(y) = M.
  + 2nd Preimage resistance: given M, find M’ such that h(M) = h(M’)
    - If you download a file, and look at its hash, compare it to the publicly known hash posted by publishers, to check it’s validity
      * The attack could generate a corrupt file with the same hash as the public one, you can attack the downloader.
  + Collision resistance: impossible to find 2 distinct messages, M and M’, such that h(M) = h(M’).
    - Difference to 2nd preimage, in 2P: you are given M and try work to find M’, whereas in Collision, you aren’t given anything but find 2 inputs that clash
* We need **I N T E G R I T Y (I)** so that we know if someone has changed the message while it’s in transit, & we need **A U T H E N T I C A T I O N (A)** so we know the endpoints of the communication line are who we think i.e. otherwise the attacker is pretending to be the bank, sending untampered but likely deceptive messages
* Side note: **Replay attacks**: *I say, “I’d like to withdraw $5000”, the bank says “Sure, he has enough in the account for that, dispense $5000”. What’s to stop me from recording that message, cutting the verification out of the communication and replaying that message saying “dispense $5000”.*
  + Idea: timestamp your message: so if you sent the original OK at 12pm, June 6, 2018 and someone replays that message at any other time, you read the stamp and know it is fake.
  + Generalised: we need something on the message that distinguishes it from all others, a unique identifier
  + Often used: nonce -> a **number** that is used **once**, mark a message with it and record all the ones you used, so if you see a duplicate, you know it’s an attack
* Establishing Integrity:
  + Parity: we establish a protocol in secret that is “every message I send you will have an odd number of words” so if someone gets a message from me that is even, they know it’s been changed
    - This will only detect, on average, half the number of attacks
    - But if someone tampers with this, half the time we will have hash collisions, when their tampered message and my normal messages can be mistaken for each other based on their hash
    - Add conditions: number of ‘e’s will be even and such to make collisions unlikely
  + In general: we take some kind of summary of the message (e.x. number of words) -> use this summary to generate some kind of hash value and the more sophisticated one’s include the hash value in the message to authenticate itself
    - Secure by obscurity -> so if someone can do your hash, this is bad
  + Take a piece of content and represent it with a number (usually huge & variable) and we map this to a small, fixed-sized summary of the content (smaller number) which we use to find the content
    - We want hashing to be evenly distributed -> mapping 1mill things to 10 hash values, we want 1000K to map to each of the values, so the chance of a collision in 1/10, not skewed
  + Hashing/Fingerprinting: making files tamper evident
    - Problem: I upload a file without malware, someone downloads it 3 days later. How do we know no-one changed it in those 3 days to include malware.
    - What we do is compute the hash of the file, convert 1 MB to 100 bits, say it’s a string of 16 characters. Then I publish this on a website of some other channel where it is publicly readable. Then someone who alters the file also needs to alter my website somehow to change the hash value to stop someone who downloads the file detecting the change. I.e. download file + compute hash must = look up website + read hash so there are extra channels that need to be compromised to cover tampering.
* **Cryptographic hash: normal hash + extra requirements.**
  + **Fixed size output for variable and changing size inputs**
  + If I work out the hash value, **I can’t generate** **a document that has that hash value by reverse engineering due to how complex the hash was.** 
    - The fastest way to crack: generate bajillion random docs, hash them 1 at a time until I get the same hash value
    - If it’s a 20-bit hash, someone has to try 2^20 things, hit it on avg 2^19 (half way though) – so someone trying to crack a cryptographic hash must do as much work as there are bits in my hash.
  + **Output must seem unrelated to input, so we cannot reverse** 
    - **Unlikely to have collisions**
  + The hash function is public knowledge, but even knowing the hash of the message, you cannot reverse it (hard to achieve)
  + They give us the ability to create a protocol that provides integrity
  + **Avalanche property: a tiny change in the content leads to a huge change in the hash value**, which is good so that similar things don’t lead to similar hash values, as this would -> non uniform distribution and ^ ability to crack
    - Changing 1 bit in the message should change half the bits in the hash
  + **Example of an application: Message Authentication Code (MAC):**
    - You and I have a small secret, the word “sausage”.
    - Add sausage to the end of all my messages, and then hash the whole thing (hash(M + sausage) to get a number, e.x. 1001.
    - I send the message and I send the message authentication code
    - From the message: cannot work out the hash value 1001, because they don’t know we used sausage
    - From the code, if you intercept: you cannot go backwards, so you can’t work out the secret
    - They can know the hash function, it will still be equivalent to brute force because of avalanche with our secret
    - Weakness – length extension attack