**Human and Weakness**

Head of the institute for corporate governance at Yale

* Governance: makes sure things are kept in control such as corrupt behaviour etc. Looking to make sure overall things are run properly.
* His role involves teaching people how to set up systems and put things in place so that organisations can run ethically, responsibilities and how to encourage this sort of behaviour.
* He was sacked for traveling on teaching expenses.

Harvard Professor

* Was in Russia, helping the govt set up honest and transparent capital markets.
* Changed with illegal speculation

Many people who are supposed to be upholding the rule of law are instead breaking it.

Auditors

* Are supposed to make sure the company they are auditing is not doing anything dodgy
* However, the auditors are paid by the company to do the auditing.
* Hence, there is a conflict of interest
* **The Enron case**

**Moral Hazard** – Someone has the incentive to act in the wrong way even though they’re supposed to act I the right way

* The bigger those incentives, the harder it is for humans to do it.
* The bigger the stakes, i.e. choose between saving the life of a friend or a stranger, humans will eventually stray and act towards their own interests.
* It is very hard to be ethical

Q&A with Police Integrity Commission

* QUESTION : How can you tell when a policeman is corrupt? Do you go look at the history if there could have been evidence, data or things that could have bene a clue to start to a form a model of how policeman go corrupt, then you work out ways next time you could detect it earlier?
* ANSWER: We’ve always looked for that, but there’s nothing we have found so far. As far as we can see, it is completely random and it could happen to everyone. Someone who’s rich/poor, a senior/junior, someone who’s happy/not happy. Banks also report the same things with their tellers.

There is a known rate where people become dishonest / corrupt inside banks. The bank again can’t find any patterns in the corruption.

**Humans eventually give in and can become corrupt. Therefore, if you rely on everyone inside an organisation keeping a secret, you’re in trouble because someone will leak it.**

If you can somehow reduce corruption by even 1% or 2%, everyone would be interested in that.

Moral Hazard start in insurance – you insure to get rid of risk, but you need the person to still wear some risk but you need the person to wear some risk, or else they would never adjust their behaviour to the risk.

People in power are very tempted to abuse that power in many ways.

**C.I.A – Confidentiality, Integrity, Authentication**

Cryptographic Analysis – We try to first work out what properties we really want, instead of jumping straight away and applying everything.

The **CIA Properties**

* **Confidentiality** – with ciphers and codes / ensuring access to sensitive information via. authorisation restrictions / encryption
* **Integrity** – we do something to the message so we know if someone has changed the message when it is travelled through the “badlands” the uncontrolled medium.
* **Authentication** – Securing the endpoints, to confirm that the message came from the stated sender (and not a fake middleman)
  + E.g. **Message Authentication Code (MAC)** is 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.
  + How a MAC works:
    - A *key generation* algorithm selects a key from the key space uniformly at random
    - A *signing algorithm* efficiently returns a tag, given the key and the message.
    - A *verifying algorithm* efficiently verifies the authenticity of the message, given the key and the tag.  
      That is, it RETURNS accepted when the message and tag are not tampered with or forged.
* **Non-Repudiation** – You can’t claim that something didn’t happen when it did.
  + E.g. We sign a contract and agree to various things. You can’t later on say “Hey I never signed that contract”
  + There is someway to verify that you did sign it which you can’t escape from.

**Replay Attacks**

* A Replay Attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or replayed to one or more of parties who may process the data as legitimate.
* “An attack on a security protocol using replay of messages from a different context to the intended context, thereby fooling the honest participants into thinking they have successfully completed the protocol run”
* How this works:
  + Alice wants to prove her identity to Bob.
  + Bob requests her password as proof of identity, which Alice complies (after some transformation like a Hash Function).
  + Meanwhile, a Spy is listening in on the conversation and keeps the password / the hash.
  + After the exchange is over, the Spy posing as Alice connects to Bob.
  + When the Spy is asked for proof of identity, the Spy sends Bob Alice’s password / hash read from the last session which Bob accepts, thus granting the Spy access.
* General countermeasure for Replay Attacks: **Timestamps / Tagging**
  + Tagging each encrypted component with a **sessionID** **and component number**.
  + A unique random sessionID is created for each run of the program therefore it is difficult to replicate the same ID in the next run. The attacker would be unable to perform the replay because on a new run, the sessionID would have changed.
* **One-time password countermeasure**:
  + Similar to **session tokens** where the password expires after it has been used or after a very short amount of time.
* **Nonces / MAC**
  + A **nonce** is an arbitrary number that may only be used once. (Hence Number Once = Nonce)
  + It is often a random / pseudo-random number issued in an Authentication Protocol to ensure that old communications cannot be reused in replay attacks. They can also be useful as **initialisation vectors / starting variable** *(a fixed-sized input to a cryptographic primate that is required to be random/pseudo-random)* or a **hash function**.
  + A nonce combined with a MAC (Message Authentication Code) should be used together which the receiver should check for any tampering.

**Hashing**

**Hash Collision**

* When two messages have the same value.
* There is no way of avoiding hash collisions, if you have a big document and is compressing into a smaller document.

Two sorts of hashing:

**1. COMPUTING HASHES**

* Take a big number (regard any message as a number). A hash summarises this big document, so we map a whole big document to a small summary of it.
* A small summary is more portable and convenient.
* Example: iTunes, if you load a song it will find the cover for your song straight away by hashes your song to find it quickly.
* In computer science, we are mainly interested in Hash Collisions. We want to have collisions distributed as evenly as possible, so that the chance of a hash collision occurring is the minimum possible chance.
* Example in Security: Compute a hash of the file in a very complex way, so that anyone that changes the file is likely to change the file.
* The hash can be represented as a 16bit character string that is visible, hence it is a lot smaller and easier to compute
* How it works:
  + When the uploader puts the file on the website, you compute the hash and print the hash on the website
  + You get the file / download it
  + Compute the hash, check it matches the hash on the website.
  + So if those two paths are the same, the attacker would have to subvert both paths
* **Hashes are an INTEGRITY CHECK**

**2. CRYPTOGRAPHIC HASHES**

* Are traditional computing hashes, but they also have a bit more requirements.
* If I know the hash value, I still can’t work out the message that has the hash value.
* For normal Computing Hashes, once I know how the hash function works, I’ll be able to work out what specific document belongs to a hash that I have.
* For a Cryptographic Hash, there is not faster way of getting a document that has your specific hash than generating a billion documents and hashing them one at a time until you get a collision.
* If you have a 20bit hash 🡪 will take you 220 tries to get a collision.
* **Cryptographic Hashes takes an arbitrary sized input, then produced a fixed sized output which is pseudo-random (there is no observation relation between the input/output, so you can’t run it backwards)**
* **Secrecy** of the cryptographic hash relies on that fact that it is really hard to undo. We make the Hash Algorithms public, so everyone knows the hash algorithms, but even know them you can’t run them backwards. It is a really good property and one that is really hard to have.
* **Hash / Message Digest** **/ Fingerprint** is the output of a Hash Function
* They are the **“Swiss Army knife of Crypto”** because you use them all the time.
  + They give us **integrity** but NOT authentication nor confidentiality
  + One-way nature. You can’t reverse or go backwards on it.
  + Variable inputs 🡪 fixed-size outputs.
  + Good at generating pseudo-random numbers.

**Telegraphic Transfer Problem**

You want to send message from A 🡪 B and want to make sure that no one fools with the message, so we use a cryptographic hash to the message.

* Don’t build a system that relies on keeping the cryptographic hash function a secret. **Security by obscurity** **itself is bad, don’t depend on it**.
* Solution: Message Authentication Code (MAC)
  + Get the message M
  + Add a secret word at the end of the message (a shared secret) (stick a key on the end of the message)
  + We now have: **M + “Secret”**
  + Compute the hash of M + “Secret” = 1001 (The MAC)
  + Send the **message AND 1001**
  + Anyone that gets this message cannot compute the hash because they don’t know the secret. They need to stick the secret to the end of the message in order to compute the hash.
* **Great property of cryptographic functions: Changing one bit in a message on average should change half the bits in the hash.** Hence making even the smallest change to the message should give a completely different hash.
* If it is a **100bit Hash, on average you will need to do 2100bits of work until you find a collision.**
* We would want about **2128bits of work**

Cryptographic Functions

* **MD5 – Message Digest 5**
  + It provides less than 64bits of security, solvable with even brute force.
* SHA2
  + Not yet cracked.
* A **crack on a hash** simply means that you can reverse it faster than brute force.
  + Even if it’s a slight improvement, such as getting it from 100bits of work to 99bits of work, then that is regarded as a breakthrough.