**RISK**

Something that most people DON’T do is weigh out risk and try to put a price on it.

* Everyone has different ways to weight it, but you NEED to do this.
* Saying that something is risky is useless, because everything will have a different relative amount of risk.
* **To be an effective security professional, you need to work out at what point, its worth taking this risk.**
* People are NOT rational risk and are not good at assessing it.
* **We need to think about risk ANALYTICALLY**

Example: the laptop repair shop with world expert catching and throwing parts.

* Maybe there is a 1% chance that they will break a laptop.
* Lets say, the shop serves 40 people and 1/40 of the laptops break.
* It’s not the fact that only 1/40 people were served badly, but 39/40 people were served badly and GOT AWAY WITH IT.
* It was just as bad what happened for those first 39 (the risk that they were put through) as the last case that broke the laptop.

Risk is INVISIBLE, because all we see are OUTCOMES. Because it is invisible, we don’t pay attention to it.

* Humans care deeply about bad outcomes, but are less observant when it comes to risk.
* If the probability is high enough for a bad thing happening, the outcomes you’ll observe are reasonable estimators of the risk. We can make sensible decisions in those cases.
* Humans are great at making decisions about risk when the chance of the bad thing happening is NOT TOO LOW.
* However, when the chance of the bad thing happening is VERY LOW yet the impact of the bad thing happening is CATASTROPHIC, these are the risks that mankind is terrible at assessing.
* WE ARE BAD AT ASSESSING LOW PROBABILITY, HIGH IMPACT EVENTS.
* Because the risk is so low, you may not even see that the risk is there until it happens first time.

In finance, risk is well-understood better than in insurance.

* How is risk priced in finance?
* Essentially, the more risk the asset has, the more reward there will be.
* E.g. If you were offered two assets, with the same expected return but one had more risk (more variance / fluctuation) then the two assets would be priced differently.
* One of the ways of bearing risk, is to shift the cost of risk to someone else.

In the financial world, risk is priced quite well. In security, this is not the case. Why not?

* The volume of trades in the financial market is huge, so if something goes wrong, it is usually something with a relative probability. Past data is a reasonable estimate of what’s going on.
* In the security world, if you’re trying to work out how to protect your firm against something that’s never happened before but is theoretically possible, you have no idea how to assess or weight that up.
* You also don’t know how much money to spend on security. You may spend too much, or too little.
* There is just not enough data to estimate risk.

People and bad assessments of risk: Volcanos

* Reasons that people who live around volcanos give:
  + Our great grandfathers have lived here and nothing has happened to them.
  + It always rumbles anyway and nothing ever happens etc.
* “We’ve never observed a catastrophe so it can never happen!”
* This is a really bad assessment of risk.
* They don’t realise that everytime it rumbles, there a risk of it exploding and that risk is the bad thing because it is CATASTROPHIC.
* You’re taking on the risk if you stay there and you’re avoiding the risk if you leave.

Another low probability high impact risk event: Pittbulls attacking children or adults in public.

United States example: Mass shooting in a school

Level crossing train tracks.

**The problem of induction**: **the past is only of limited use in predicting the future**.

It is really good for predicting the future for high-probability events.

For low-probability events, the risk is invisible. We can’t see it.

Data-breach notifications

* America’s companies have mandatory reporting for all data-breaches + announcements on stock exchanges too, which CEO’s hate.
* In Australia, we do not have compulsory data-breach notifications.

Centralising services

* We always store things in the cloud these days
* Computing things together on AWS cloud
* Instead of using different mailservers, use Gmail.
* Instead of using different operating systems, use OSX, Microsoft.
* Everyone used to use different streaming services, now there is only a few.
* When things become centralised, you get an economies of scale, giving a network effect.
* **Because everything is becoming more centralised, the consequences can be catastrophic.**
* **Although companies such as Google can increase the resources to decrease the chance of something happening, they’re increasing the potential impact of something happening.**

Other examples of low-probability high-impact things:

* Dinosaurs all had similar build, being big etc. When a meteor struck Earth, it catastrophically wiped them all out because they were all concentrated/ similar.
* Agricultural industry everywhere has same types / species of crops, fruits etc. When one disease hits, then it can wipe the rest out catastrophically.

**PUBLIC KEY CRYPTOGRAPHY**

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.

Merkle (earlier attempts to solve keys problem)

* Merkle was a student, who dreamt of a way to send a message to someone without having shared a key first and the recipient is the only one who could read the message.
* Method:
  + Merkle writes a series of notes. Each note has a different message on it + different number.
  + The messages follow as: “I am number 1. My key is sausage” “I am number 2. My key is pants”.
  + These messages are encrypted first, enciphered with a different key.
  + He takes all these encrypted messages and publishes it on the internet, so everyone can see the cipher texts.
  + We want to talk to Merkle, so we find one of the bits of ciphertext and we pick that one.
  + We try for a brute force deciper method. Let’s say we crack it in 1 minute.
  + When it is cracked, we can read the message containing the key.s
  + We say to him, “here is a confidential message from me”, I am using “Number 2, with key Pants”.
  + When Merkle receives this message, he knows what’s written on every bit of paper, so he goes to paper Number 2 and decode the message straight away.
  + However, anyone that intercepts the message and tries to decode it will have to crack the code on every piece of paper until they crack the message corresponding to paper Number 2.
  + On average, they’ll have to look at half the pieces of paper.
  + **Basically this security is built on the fact that the two-parties are using ONE SELECTED KEY to message each other. If anyone finds the encrypted message and tries to crack it, they won’t know which key was selected in the beginning and will have to crack each key until it matches.**
* You essentially want to be able to encode in constant time / want decoding to take an awfully long time when someone is trying to brute force.
* A whole system has to be kept secret to make it work
  + Usually we set up a system and keep using it.
  + If we rely on the program staying secret, it will be broken
* We can compress all the secrecy in a key.

**DISCRETE LOG PROBLEM**

xy % z = 100,101

* If we’re given x, z and the result, it is extremely difficult to work out y.
* Trying to figure out the **y** is the **Discrete Log Problem**

Example: If we’re given say 517 x 624

* Multiplying two numbers together to get the result is fast.
* However, if we’re given a resulting number and we need to figure out the factors, it will be slow.

Some cryptographic systems, such as RSA relies on the fact that multiplication is fast, but undoing it is slow.

The asymmetry of these two calculations is what drives public key cryptography.

* One is fast to calculate (getting the result)
* The other is slow to calculate (finding the factors / undoing the number)

If mathematicians can figure out a way of solving the Discrete Log Problem faster, then everyone would be interested in knowing this.

* The NSA would be interested
* Results might not even get published, because anyone that can solve the DLP faster than anyone else is now better at codebreaking.

**Diffie-Hellman Key Exchange (Symmetric key cryptography)**

How it works

* Background
  + Early stage encryption required two parties to share a secret random number, a key.
  + How do two people who have never met agree on a secret shared key, without letting someone who is listening also obtain a copy?
* The Diffie-Hellman Key Exchange = A key-exchange algorithm to establish a shared symmetric key.
  + The trick is based on two facts. Image the keys as colours:  
    1. It is easy to mix two colours together to make a third colour  
    2. Given a mixed colour, it is hard to reverse it to find the exact original colours
  + This is the basis for a lock: easy in one direction but hard in the reverse direction.
  + This is know as a **one-way function**
* In a numerical perspective
  + Easy in one direction, hard in the other direction
  + We use Modular Arithmetic to do this.
  + Congruency Example:
    - If we calculate 46 % 12 = 10
    - We say that **46 % 12** **is CONGRUENT to 10**
  + Numerical Example:
    - We use a **prime modulus (prime number mod)** **such as 17**
    - Then we find a **primitive root of 17 which is 3**, which has an important property that when raised to different exponents, the solution distributes uniformly around the clock.
    - **3 = the generator**. If we raised 3 to any exponent X (i.e. 3X) the solution is likely to be any integer between 0 and 17.
    - However, the reverse is difficult. Given the integer 12, find the exponent X that 3 needs to be raised to = **The Discrete Logarithm** function.
  + This numerical example is the one-way function, easy to perform but hard to reverse.
  + Given 12, we would have to **trial and error** to find matching exponents X.
    - How difficult is it to do?
    - Given a SMALL PRIME MOD, this would be easy.
    - However, given a LARGE PRIME MOD which is hundreds of digits long, it becomes impractical to solve even if we had access to all computational power on earth, it could take thousands of years to run through all possibilities.
  + The strength of this one-way function is based on the time needed to reverse the function.
* Using Alice/Bob example with the one-way-function **gX % m = p** :
  + Both Alice/Bob agree PUBLICLY on a PRIME MODULUS (P) and a GENERATOR (G)
  + Alice selects a PRIVATE RANDOM NUMBER (X = 15) say 15: **315 % 17 ≡ 6**
  + **The result 6 is sent PUBICLY to Bob.**
  + Bob selects his PRIVATE RANDOM NUMBER (X = 13) say 13: **313 % 17 ≡ 12**
  + **This result 12 is sent PUBLICLY TO ALICE.**
  + Alice takes Bob’s public result and raises it to her X to obtain the SHARED SECRET **1215 % 17 ≡ 10**
  + Bob takes Alice’s public result and raises it to his X to obtain the SHARED SECRET **613 % 17 ≡ 10**
  + **Now they both have the same shared key and can be used as a SYMMETRIC KEY  
    (can be used for both Encryption / Decryption)**
* How do they both arrive at the same key?
  + This is to do with the property **xa^b = xa\*b** ⇔ **xb^a = xb\*a**
  + Technically, Alice is calculating: **313^15 = 313\*15** for the modulus calculation
  + Meanwhile, Bob is calculating: **315^13 = 315\*13** for the modulus calculation
  + Hence they are doing the exact same calculation.
* Without access to one of the PRIVATE KEYS 13 / 15, the spy would not be able to find the solution.
* It is practically impossible for the spy to break the encryption in a reasonable amount of time.
* It can be used in conjunction with a **pseudorandom number generator** to encrypt messages between people who have never met.
  + A pseudorandom number generator is an algorithm for generating a sequence of of numbers whose properties approximate the properties of sequences of random numbers.
  + PNG’s are **not truly random** and is a **deterministic algorithm (algo which given a particular input, will always produce the same output)** determined by a relatively small set of initial values **(the seed)**.

**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.**