

# Cryptographic Protocols

Jack Bradbrook

# Topics

- The Internet Protocol Stack
- Common non secure protocols: HTTP, TCP, UDP, IP
- Tunnelling Protocols
  - IPSec
  - SSL
- Encapsulating & carrier protocols
- VPNS
- States of Data
- Key Escrow
- Mobile Telecommunications (GSM)

# Aim of this lecture

- We need to look at how the methods we've explored apply to some real protocols according to the module Spec:

- 1.3 Show where the various cryptographic techniques may be employed to secure data and systems. For example, but not limited to:
  - Password verification
  - Digital signatures
  - VPNs
  - Tunnelling
  - Encapsulating & carrier protocols
  - IPsec

# What is a communication protocol?

- “In telecommunication, a **communication protocol** is a system of rules that allow two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both”

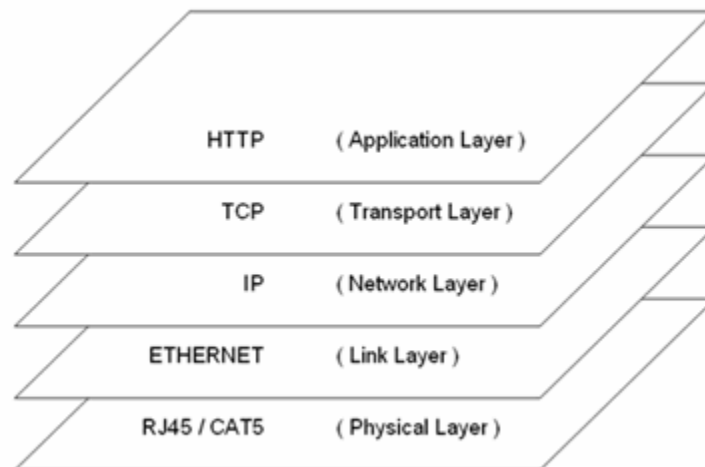
Question: Why do we need communication protocols?

# Why We have Protocols?

- Protocols allow information to be shared between devices in different geographic locations
- Protocols allow devices which are very different to communicate with one another in a way they can both understand.
  - Protocols mean we can have a super computer with a vast amount of resources and a complex architecture communicating with a low power mobile device using a basic architecture.

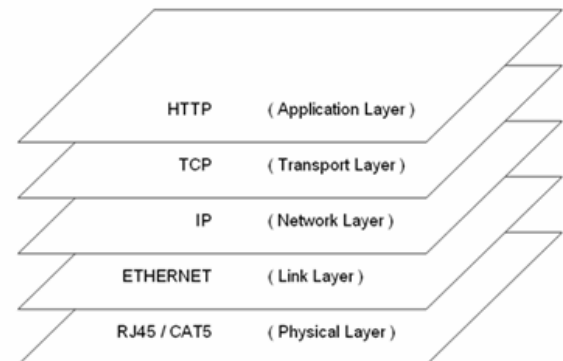
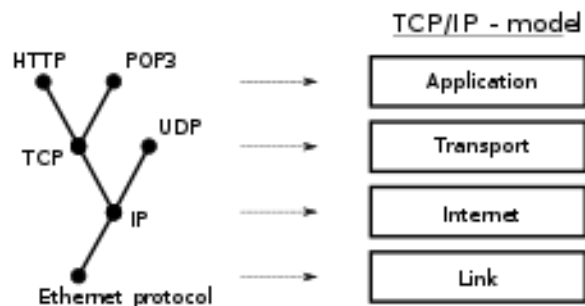
# Internet Communication

- Do people know about:
  - Internet Protocol (IP)
  - Transmission Control Protocol (TCP)
  - User Datagram Protocol (UDP)



# The Internet Protocol Stack

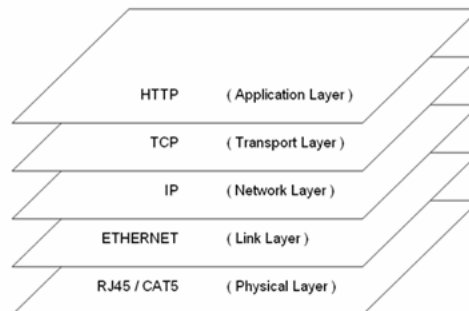
- Protocols work at different levels
- Different Protocols have different responsibilities and capability and are selected depending on the task they are needed for
- The higher level protocols add extra functionality which makes performing more complex tasks easier.
- Higher level protocols are typically designed specific uses, lower level protocols are there to solve more fundamental problems with how communication occurs.
- The higher level protocols make use of lower level protocols to perform simple tasks on their behalf





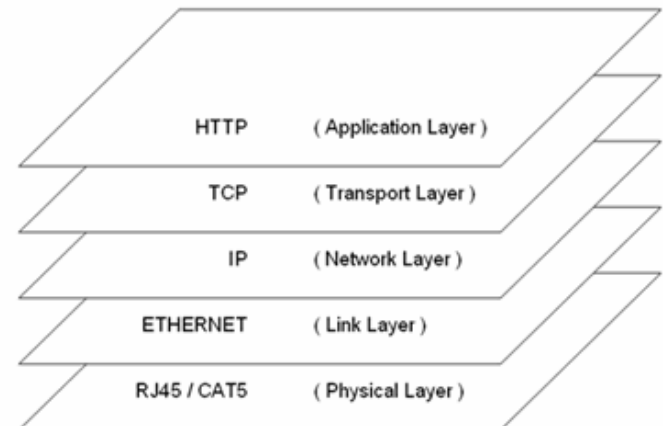
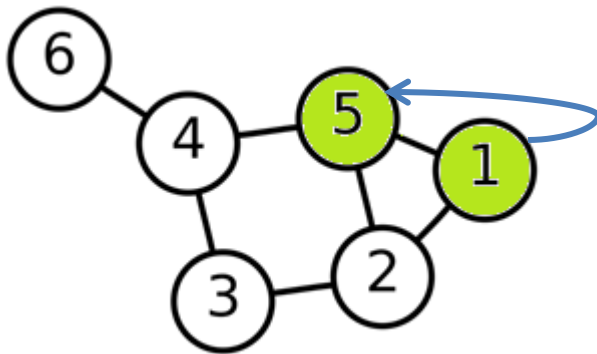
# The Physical Layer

- The physical layer is at the bottom of the protocol stack
- RJ45/ CAT5
- It specifies the physical medium with which we can send our messages over, without this we can't do anything, there is no communication system.
- E.g. What kind of cable can we use for communication?



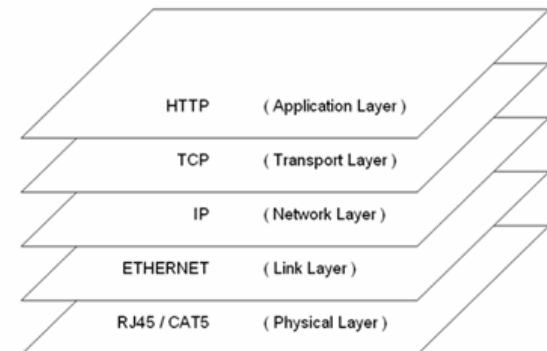
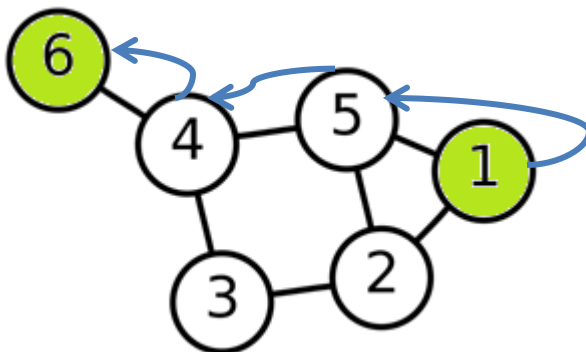
# Link Layer

- The link layer is the lowest software layer in the protocol stack.
- The network layer takes a packet given to it by the higher layers and arranges it into a series of bits which are what is sent using the physical layer.
- Unlike the higher layers in the stack the link later only deals with direct communication between two machines in the network, when communication is done across many machines that is handled by the Network layer.
  - E.g. computer 1 can send a message to computers 2 and 5 using the link layer but not 3, 4 or 6



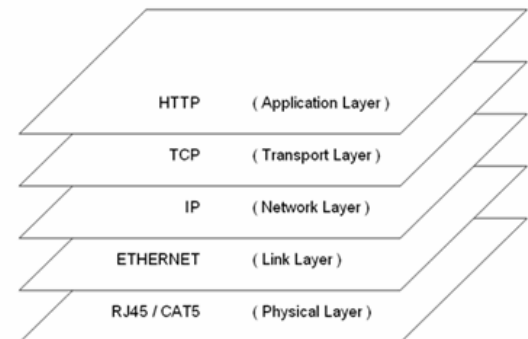
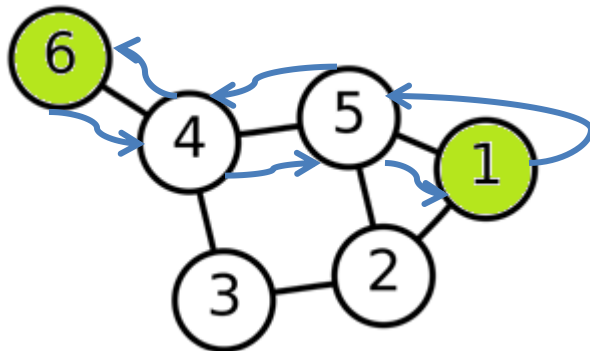
# Network Layer

- The network layer is responsible for transporting a single message from one machine to another.
- The message is split into data packets and each packet is routed across the network via routers.
- This means there does not need to be a direct connection between the two machines.
- E.g. machine 1 can send a message to machine 6 using the network layer.
- This is because machines 2, 3, 4 and 5 are also part of the system and route the message towards the machine which has it's destination address.



# Transport Layer

- The transport layer is about allowing machines to establish a channel of communication in order to send larger volumes of data to one another.
- In the case of TCP this is used to provide error resistant communication between the machines.
- If I'm machine 1 wants to send a message to machine 6 and be sure that the message has received that message I need to use a transport layer protocol
- Backward and forward communication needs to be established between machines 1 and 6 so machine 6 can tell machine 1 that it has received all the packets that make up a message.



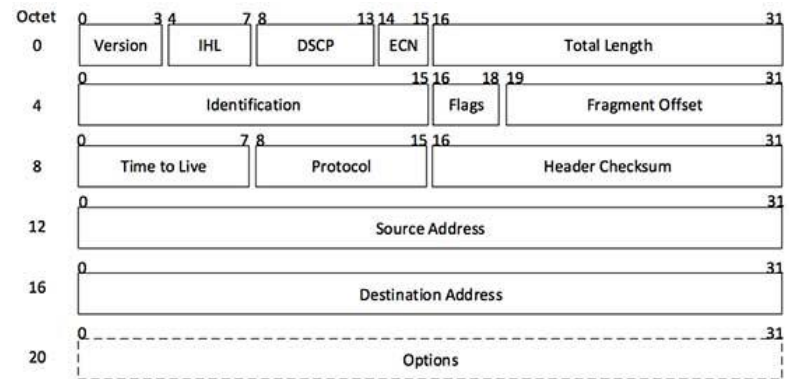
# Application Layer

- This is the highest layer of the protocol stack.
- It's the layer that our specific software applications interface with.
- Application layer protocols provide specific units of functionality useful to software applications for performing various tasks using the lower level protocols.
- E.g. HTTP (Hyper Text Transfer Protocol), FTP (File Transfer Protocol),
- Think about when you access a website, you never have to tell the browser to individually send IP packets down a cable and check that they've arrived, you ask the web browser to go to a website and it makes the request to get that website using the application layer (HTTP), which in turn uses the transport layer and so on...



# Packets of Data

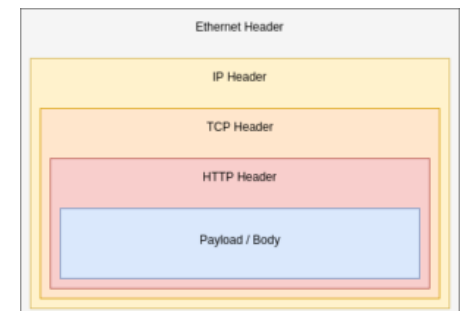
- Data is sent over a network in packets.
- Packets are made up of:
- A Header – contains information allowing the communication of the packet to happen e.g. source and destination address, protocol type.
  - Without this the software implementing the protocol wont be able to do anything with the packet.
- A body – The information which the application or user wants to communicate



[Image: IP Header]

# Packet Encapsulation

- Packets sent over networks look different depending on which protocol they are associated with, an IP packet is different to a TCP Packet which is different to a HTTP packet.
- Higher level protocols need bigger packets which contain more header information.
- Protocols at each layer of the stack need the information required for communication to be performed at their level for that packet.
- Packets for higher level protocols still need the functionality provided by lower level packets.
- Packets for higher level protocols therefore encapsulate the headers of the packets for the lower level protocols.



# Internet Protocol (IP)

- The internet protocol is the core protocol used by all communications that happen over the internet.
- Any data sent between different devices over the internet uses IP
- It's task is solely to get a packet from one computer to another computer.



(IP Encapsulation)

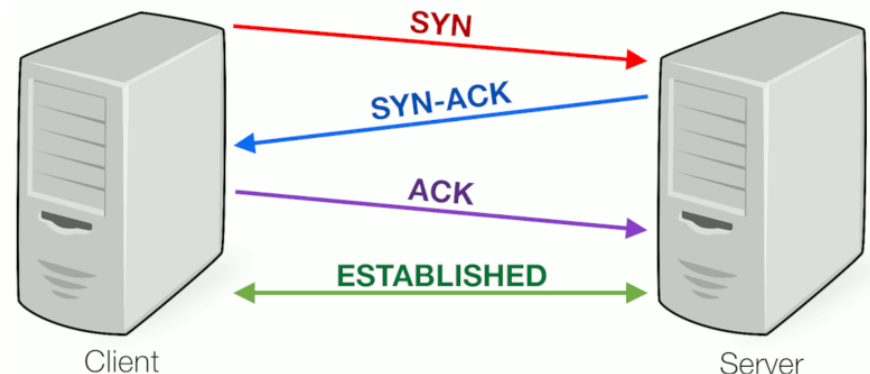


# Transmission Control Protocol (TCP)

- Transport layer protocol (1 layer up from IP)
- Establishes a channel of communication over a IP network.
- IP is only used for sending a single message from a sender to a receiver, what if we want to establish backward and forward communication between two applications?
- If we want to send a message with more than one packet of data we need to make sure that all the packets arrive at the other end.
  - Imagine going to a website and only seeing half the page because the packets were lost for the other half.

# The TCP handshake

- TCP establishes a connection between the sender and receiver using something called the **Three Way Handshake** this allows both machines to know they can successfully communicate with one another before more packets are sent.
- The **three way handshake** consists of a communication involving three types of messages:
  - **Syn** – The client sends a SYNchronize message to the recipient
  - **Syn-Ack** – The server sends a SYNchronization ACKnowledged message to indicate it received the SYN
  - **Ack** - The client sends an ACKnowledge back again on recieval of SYN-ACK

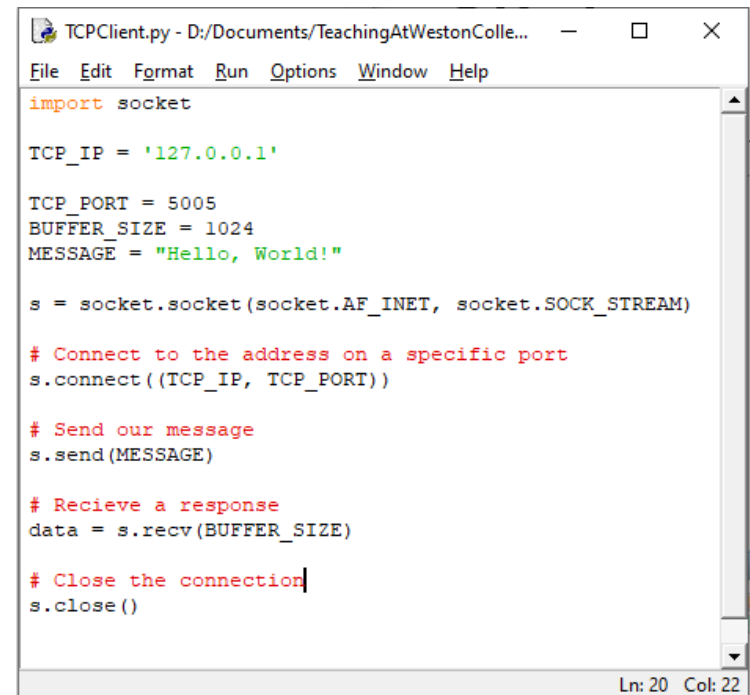


# TCP

- TCP allows for reliable transmission of packets from one party to another
- It establishes a “connection” between two computers
- Software using TCP asks the protocol to open a connection to an IP address. send and receive data to that IP address then close the connection
- TCP splits this data into packets on their behalf.

# TCP Code in Python

- Note I'm just sending a specific message as a string, there's no notion of transferring a file or anything more complex.
- Once the message has been sent I wait to receive a response from the TCP server.

A screenshot of a code editor window titled 'TCPClient.py - D:/Documents/TeachingAtWestonColle...'. The window contains Python code for a TCP client. The code imports the 'socket' module, defines 'TCP\_IP' as '127.0.0.1', 'TCP\_PORT' as 5005, 'BUFFER\_SIZE' as 1024, and 'MESSAGE' as 'Hello, World!'. It then creates a socket object 's' using 'socket.AF\_INET' and 'socket.SOCK\_STREAM', connects it to the specified IP and port, sends the message, receives a response, and finally closes the connection. The status bar at the bottom right shows 'Ln: 20 Col: 22'.

```
import socket

TCP_IP = '127.0.0.1'

TCP_PORT = 5005
BUFFER_SIZE = 1024
MESSAGE = "Hello, World!"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the address on a specific port
s.connect((TCP_IP, TCP_PORT))

# Send our message
s.send(MESSAGE)

# Recieve a response
data = s.recv(BUFFER_SIZE)

# Close the connection
s.close()
```

# User Datagram Protocol (UDP)

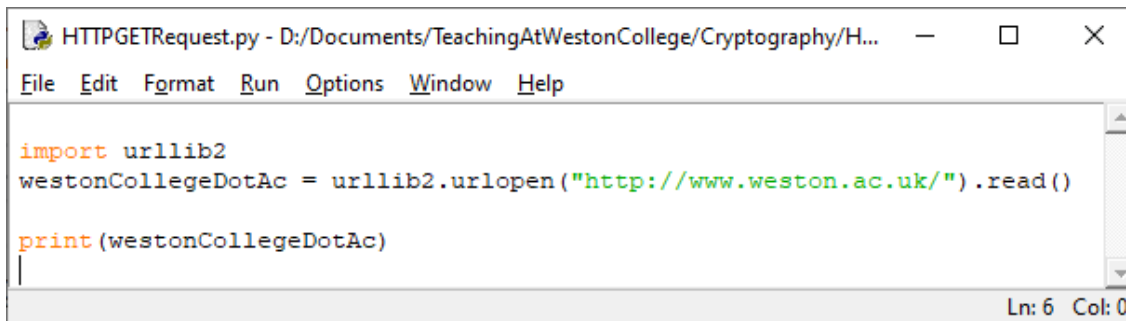
- Sends data as a stream without any methods to check whether packets arrive or not.
  - This make is a very fast way to communicate lots of data.
- No reassurances about data successfully making it to target.
- The receiving PC needs to be ready to receive all the packets within quick succession.
- Used where we need continuous data transmission but don't care if a part of the transmission is lost.
  - Live video Streaming.
  - Computer Games.

# Hypertext Transfer Protocol (HTTP)

- Application layer Protocol.
- Instead of thinking in terms of packets or messages we want to ask for files from other computers.
- This is how the WWW works
  - Computer A asks Computer B for a webpage which is just one or several files. Under the surface HTTP does this by allowing computer A to establish a connection with Computer B and download the file as packets. Each packet is then sent using IP.
- When I type the following into the address bar of my browser:  
<http://www.weston.ac.uk/news-and-events/news>
- What we're doing is telling the browser: "Use the HTTP Protocol to go to the domain [www.weston.ac.uk](http://www.weston.ac.uk) after that search for a sub domain called "news-and-events" under that if there is a page called "news" then return that to me"

# HTTP Python Example

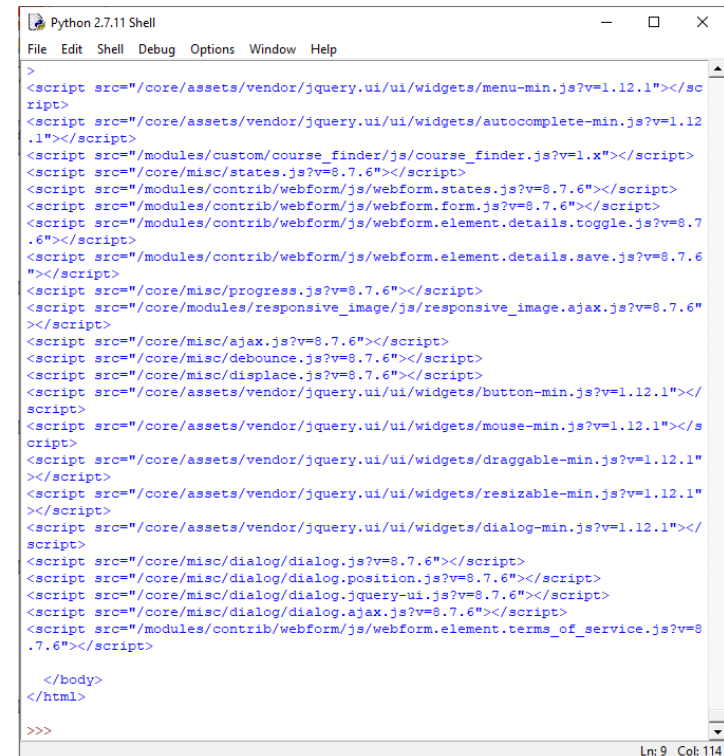
- Simple request structure, obviously to show the webpage I need some kind of browser which converts the HTML, CSS and JavaScript code into a website “[www.weston.ac.uk/](http://www.weston.ac.uk/)”



```
import urllib2
westonCollegeDotAc = urllib2.urlopen("http://www.weston.ac.uk/").read()

print(westonCollegeDotAc)
```

Ln: 6 Col: 0



```
>
<script src="/core/assets/vendor/jquery.ui/ui/widgets/menu-min.js?v=1.12.1"></script>
<script src="/core/assets/vendor/jquery.ui/ui/widgets/autocomplete-min.js?v=1.12.1"></script>
<script src="/core/misc/states.js?v=8.7.6"></script>
<script src="/modules/custom/course_finder/js/course_finder.js?v=1.x"></script>
<script src="/modules/contrib/webform/js/webform.states.js?v=8.7.6"></script>
<script src="/modules/contrib/webform/js/webform.form.js?v=8.7.6"></script>
<script src="/modules/contrib/webform/js/webform.element.details.toggle.js?v=8.7.6"></script>
<script src="/modules/contrib/webform/js/webform.element.details.save.js?v=8.7.6"></script>
<script src="/core/misc/progress.js?v=8.7.6"></script>
<script src="/core/modules/responsive_image/js/responsive_image.ajax.js?v=8.7.6"></script>
<script src="/core/misc/ajax.js?v=8.7.6"></script>
<script src="/core/misc/debounce.js?v=8.7.6"></script>
<script src="/core/misc/displace.js?v=8.7.6"></script>
<script src="/core/assets/vendor/jquery.ui/ui/widgets/button-min.js?v=1.12.1"></script>
<script src="/core/assets/vendor/jquery.ui/ui/widgets/mouse-min.js?v=1.12.1"></script>
<script src="/core/assets/vendor/jquery.ui/ui/widgets/draggable-min.js?v=1.12.1"></script>
<script src="/core/assets/vendor/jquery.ui/ui/widgets/resizable-min.js?v=1.12.1"></script>
<script src="/core/assets/vendor/jquery.ui/ui/widgets/dialog-min.js?v=1.12.1"></script>
<script src="/core/misc/dialog/dialog.js?v=8.7.6"></script>
<script src="/core/misc/dialog/dialog.position.js?v=8.7.6"></script>
<script src="/core/misc/dialog/dialog.jquery-ui.js?v=8.7.6"></script>
<script src="/core/misc/dialog/dialog.ajax.js?v=8.7.6"></script>
<script src="/modules/contrib/webform/js/webform.element.terms_of_service.js?v=8.7.6"></script>

</body>
</html>

>>>
```

Ln: 9 Col: 114

# Security and the Protocol Stack

- The protocol stack by default has no notion of security.
- You can send a message using an application level protocol and it may be completely unencrypted.
- Security can be added if chosen at each different layers of the stack by using protocols such as:
  - IPSec protocols (Internet Layer)
  - SSL (Transport Layer)
  - HTTPS (Application Layer)



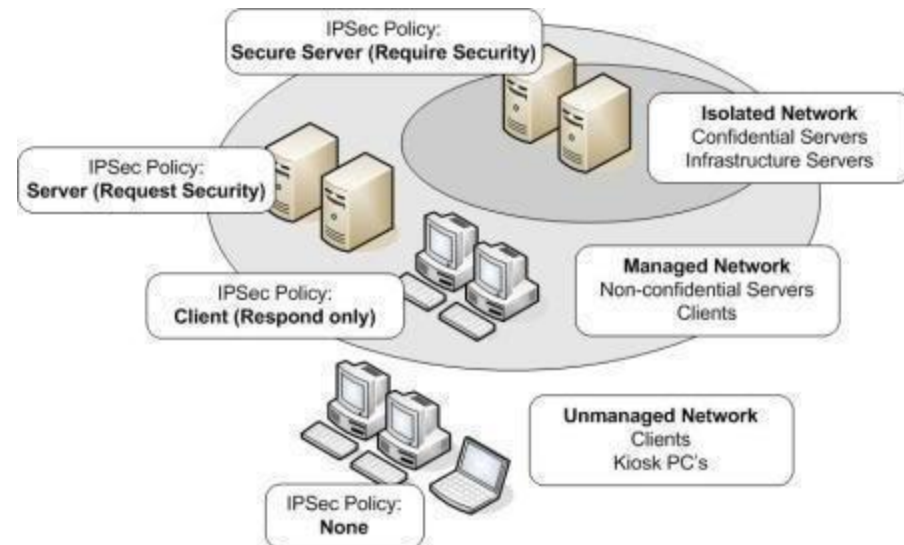
# Secure Protocols

# IPSec

- IPSec is a *“suite of protocols that allow secure encrypted communication between two machines over an unsecured network”*
- IPSec does this by applying extra security to packets that need to be communicated with particular parties that require secure communication.
- Think of IPSec as more of a system for managing secure protocols at the Internet layer, IPSec is responsible for deciding how and when these protocols should be used.

# IPSec Policies

- IPSec is configured by IPSec policies
- An IPSec policy specifies the following:
  - The Type of traffic IPSec Examines – what data is encrypted.
  - How the traffic is secured by IPSec – what protocol should be used to secure the data.
  - The Authentication method – how the system should validate that the other party is who they say they are
- There are two default IPSec policies:
  - The Server (Request Security)
  - The Client (Respond Only)



# Server (Request Security)

- This policy specifies that all IP traffic with the machine must be secured using IPSec.
- If I'm a server such as a server hosting an online banking system I don't want to allow people to do their banking with me without the packets being authenticated and encrypted.
- This policy can therefore be used to force all other computers communicating with me to use extra security when talking to the server.

# Client (Respond Only)

- When a computer uses this policy it will only use IPSec security features when the computer it is communicating with requests it to be used.
- For example I want to do online banking if the bank insists I apply security to my IP packets I will do so but if I'm accessing a regular website perhaps I don't need to add extra security.

# The IPSec Protocols

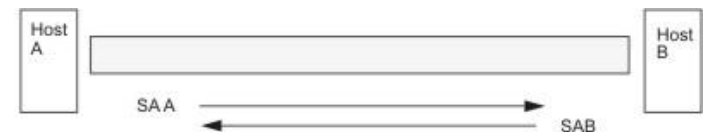
- IPSec has specific protocols in order to add different forms of security to the data being communicated.
- It's beneficial to have multiple security protocols at the internet level as any operation performed to secure the data is going to be performed for every packet.
  - The potential slowdown by adding too much security to a large volume of data is significant
- The two protocols which come with IPSec are:
  - Authentication Headers (AH)
  - Encapsulating Security Payload (ESP)
- IP Sec works in two modes
  - Transport Mode
  - Tunnel Mode
- Depending on the Mode an IPSec Packet can differ in how it looks (we'll get onto this in a bit)
- In both modes however IPSec packets are IP packets encapsulated within additional headers.

# Complexity of managing IPSec

- Using IPSec there might be many other computers that we are communicating with using a bunch of different settings such as: hashing algorithms, public keys, encryption algorithms, IPSec protocols (AH or ESP) etc.
- We need a way to keep track of what security settings we are using for each machine that we are connecting with
- That way we don't have to re establish the settings each time we want to do some communication

# Security Associations

- Security Associations (SA) are a management feature of IPSec, they form part of the architecture that AH and ESP work under.
- Using IPSec both parties make a database of security associations.
- Each Security Association in the database stores information about how that individual party should communicate with other machines.
- Each record in the database contains the following information:
  - Security parameter Index – a Unique ID associated with the SA
  - The IP destination address – An IP address that we are using IPSec to communicate with.
  - The settings used (hashing algorithms, public keys, encryption algorithms, IPSec protocols (AH or ESP))
- Allows a computer to carry on where it left off using the settings it had last time communication happened



SA = Security Association, consisting of:

- Destination address
- SPI
- Key
- Crypto Algorithm and Format
- Authentication Algorithm
- Key Lifetime



# Internet Key Exchange (IKE)

- When setting up a SA the Internet Key Exchange (IKE) needs to be performed.
- IPSec needs to establish something called the **Policy Set** before communication can start. The policy set is the information both parties are going to use in order to add security to their data when communicating.
- The **Policy Set** includes:
  - Hashing: The hashing Algorithms we're going to use e.g. SHA1, MD5
  - Authentication: what method do we want to use for authentication?
    - We've seen digital signatures, there are other methods but digital signatures are by far the most common
  - Group (Diffie-Hellman): What public keys are we going to use for our asymmetric encryption in order to share our symmetric keys
  - Lifetime: How long our communication is going to last for before we close it down and restart it.
  - Encryption: What symmetric algorithm are we going to use to encrypt our data, (typically a well known block cipher)

# Authentication Header

- The more basic of the two IPSec Protocols
- Using the AH protocol will provide authentication to data which is sent using IPSec and the IP header containing to send and receive location.
- Remember hashing and digital signatures?
- Using AH we essentially “Sign” each of our packets.  
See end of “04CryptographyInTheDigitalWorld” if this is unclear
- This means that the other party knows the following:
  - We were definitely the person who sent the message.
  - Nobody in the middle has interfered with the message (Including the IP header)
  - The message was initially intended to be sent to them not somebody else and re directed.

# Encapsulation Security Payload (ESP)

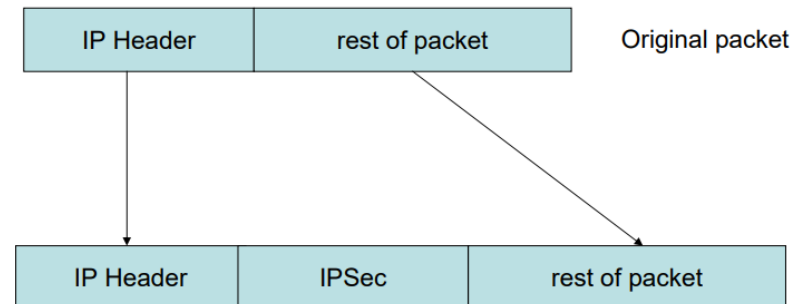
- The Encapsulation Security Payload is the main protocol IPSec uses.
- Unlike AH it encrypts the data.
- It does not authenticate the header however which is why AH hasn't been completely replaced by ESP.

# Modes In IPSec

- Transport Mode
  - Provides end to end security between two machines.
- Tunnel Mode
  - Provides security over a section of a the route between two machines but can be used for entire path, e.g. communication between two firewalls guarding private networks.

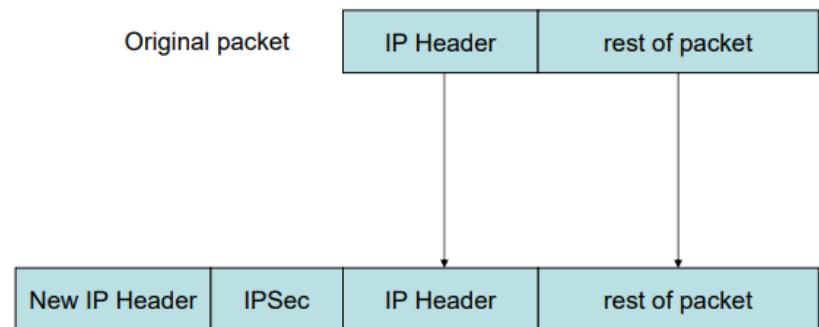
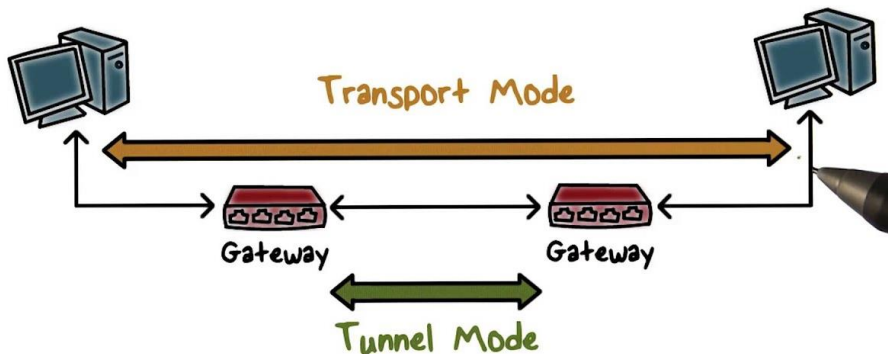
# Transport Mode

- Add additional IPSec Header between the IP Header and the payload.
- Headers are only removed at the very end of the communication.



# Tunnel Mode

- In Tunnel Mode The original packet is fully encapsulated and a new IP + IPSec header is added on at the start of the tunnel.
- When the packet reaches the other end of the tunnel the new IP Header and the IPSec header are removed and the original IP packet continues its journey without security as it would normally do if there was no IPSec.
- This means IPSec only exists for the duration of the tunnel in the communication.



# Secure Sockets Layer (SSL)

- Has for the most part replaced IPSec
- Is a form of Transport Layer Security (TLS)
- SSL exists to allow you to make sure that the computer you are talking to is one you trust!
- When doing online banking you have to be absolutely confident you are talking to the real server at the bank.

# SSL Overview

- Performs a handshake between the two communicating machines before communication begins (Similar to TCP)
  1. Computers agree on how to encrypt their data.
  2. Server sends a certificate and public key to Client to perform encryption with.
  3. The client requests encryption to begin.
  4. The server confirms that encryption is ready to start.
  5. Messages between the client and the server are then encrypted.

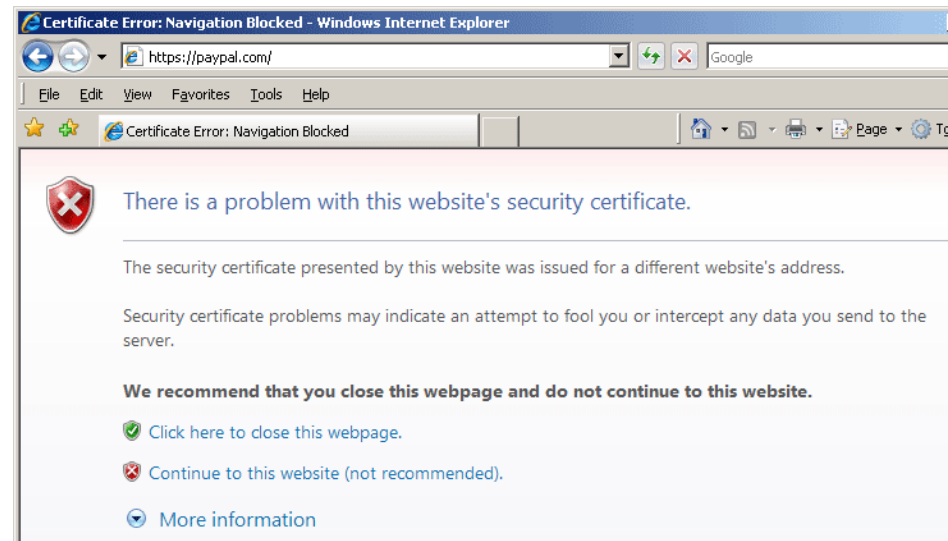


# SLL 1: Agreement on Encryption

- The machines first agree on three things:
  1. The Asymmetric system used to share the keys e.g. Diffie-Hellman, there's also some other systems (Look up RSA and DSA) (See 03AsymmetricCryptography if unsure about this)
  2. The cipher used to encrypt the data (A specific block Cypher, will look at some examples of these in next lecture)
  3. A Hash function both parties will use to generate a **Message Authentication Code** (SHA/ MD5), this provides integrity of the messages (See digital signature + examples in 04CryptographyInTheDigitalWorld if these concepts are unclear)

# SSL2: The Certificate

- Before communication begins using SSL the server sends a certificate to the client.
- The purpose of the certificate is to link a servers public key to their real identity.
- Why do we need this?
- What does it mean?



# Why Certificates Exist

- We can use Digital Signatures to make sure that nobody in the middle has tampered with the message when communicating.
- We can't use digital signatures to know for certain that the person at the other end is 100% who they say they are however.
  - The man in the middle could still hash the messages and produce a valid digital signature.
- We're still vulnerable to a man in the middle attack.

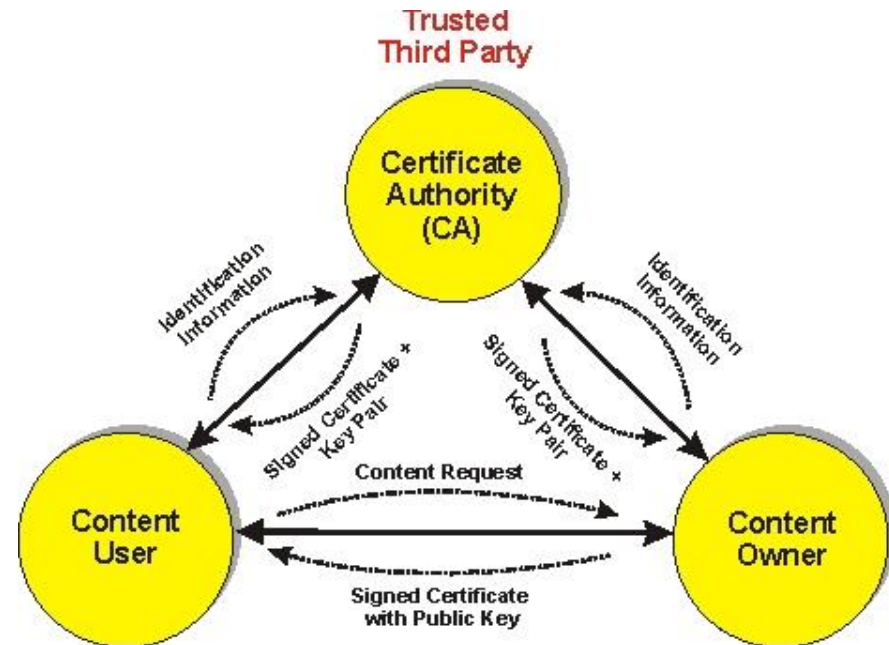
# The Certificate

- A **Digital Certificate** is a file that has been provided to the owner of the server by a **trusted authority**.
- A server can only be allocated a certificate if it applies for one, it has to provide the **trusted authority** with information about the:
  - Web Server
  - What is the purpose is of the organisation running the server
  - Where the organisation running the server is based.
  - The physical location of the server
- The certificate will contain:
  - Who allocated the certificate.
  - The name of the person or service who received the certificate.
  - An Expiration Date for the certificate.
  - A **digital signature** provided by the **trusted authority**,  
(This is important to show the certificate hasn't been tampered with) (Integrity)



# Certificate Authority Diagram

- The client receives a certificate then checks the digital signature of the certificate with the trusted certificate authority who allocated it.
- Think of a certificate like a passport for a server:
  - You apply to the government for a passport. The government checks your identity and provides you with one. This means you can use your passport to prove who you are to other people.
  - If someone doubts your passport they can check with the government and the government will be able to check if the passport is real or a fake.

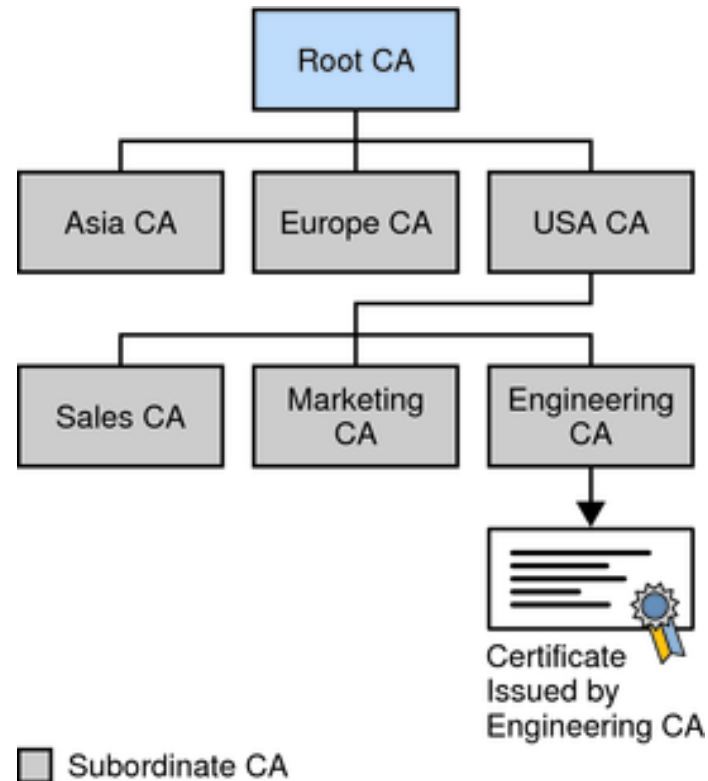


# Trusting the Authority?

- This is all great, but how do we know when we check the certificate is from the authority that we can trust the authority?
- How do we know that the authority itself isn't also a man in the middle attack?
- There are many Certificate authorities and they are run by many people, how are we supposed to trust all of them?

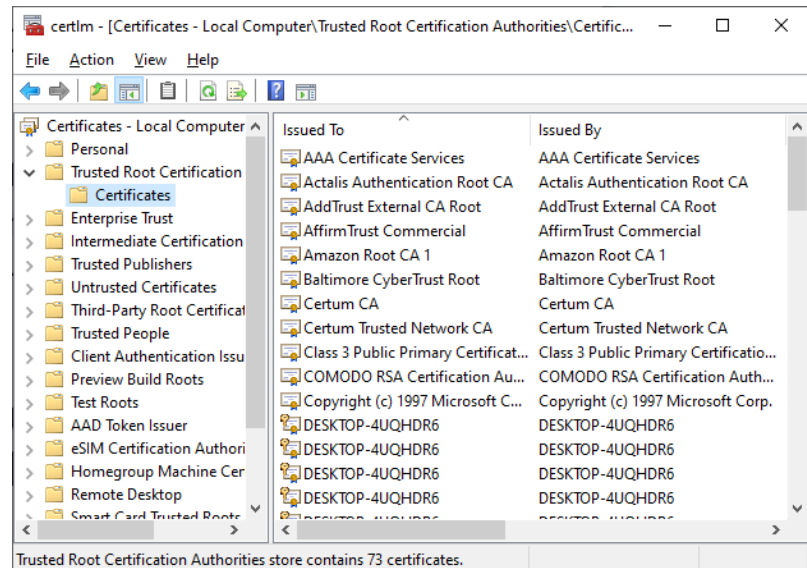
# Authority Tree

- All certificate authorities have to get their own certificates from an original “Root Certificate authority” via the same process
- If you can trust the Root Authority then you can trust the authorities underneath it.
- So now all we need to know now is how can we trust the root authority?



# Trusting the Root

- On your computer there will be files which came with the operating system e.g. Microsoft Windows which contain certificates for the root authorities.
- This means when you buy a computer or install an operating system your trust in that operating system carries over to trusting the root authorities.
- You've accepted that in so far as the software you have chosen to use doesn't have hidden malicious intentions the root authorities also aren't malicious.



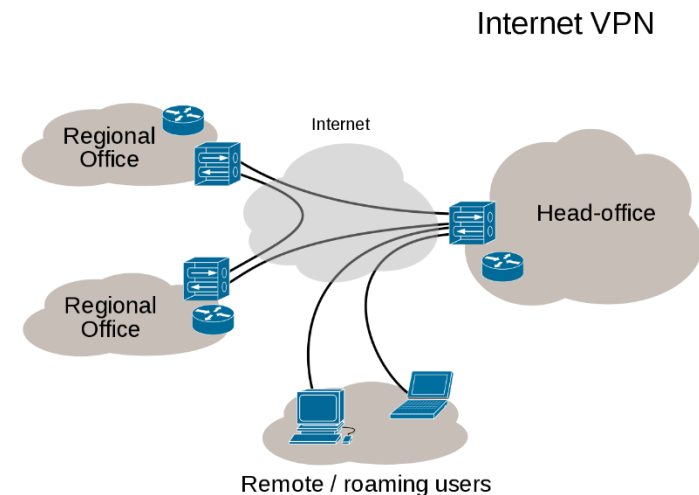


# SSL 3:

- Both machines have now agreed the settings for their communication
- The client has the ability to trust the server
- The asymmetric key exchange process now occurs using for example Diffie-Hellman (See 03AsymmetricCryptography)
- Both machines are then ready to start communicating. using a symmetric encryption method.

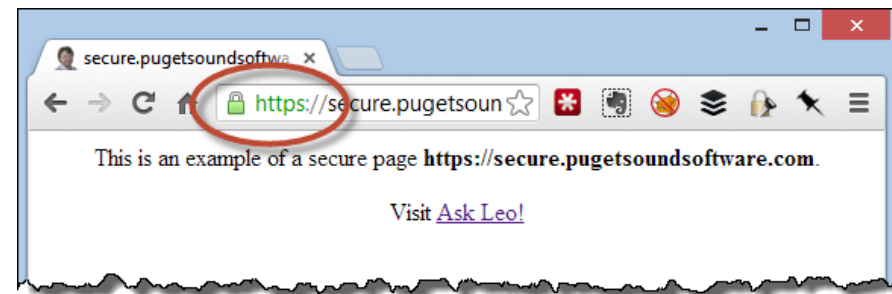
# Virtual Private Network (VPN)

- The idea: have two computers in different places on a public network connected as if they are both part of the same private network
- Allows a companies IT infrastructure to be physically distributed but behave virtually as a singular unit.
- VPN Systems use Tunnelling provided by the IPSec or SSL/TSL protocols in Order to encrypt traffic over the Internet.



# Secure Hypertext Transfer Protocol (HTTPS)

- HTTPS uses SSL or another protocol called TLS (the successor of SSL)
- Many Websites now used HTTPS by default, Google now marks websites which do not use SSL to encrypt their traffic to the user as “not secure” and demotes them in their search.
- Same kind of thing as regular HTTP, I can GET a webpage only the web page data is encrypted using SSL.



# Hash-based Message Authentication Code HMAC

- Another authentication system similar to Digital Signatures.
- A client and a server share a symmetric key using an asymmetric system.
- The client hashes their symmetric key with the message they are sending to the server. This is the HMAC.
- The server can read the message, combine it with its copy of the symmetric key and hash that to create its own HMAC
- The Server can then Check if the two HMAC values are the same

# States of Data

- Data is described as being in several different States:
- At-Rest
  - Data is stored somewhere not being used.
- In-Motion
  - Data is being transferred.
- In-Use
  - Data is being used by a person or piece of software.

# Easy Exercise

- Which of the following is: Data at Rest, Data in use, or Data in Motion?
  1. HTTPS traffic
  2. Data Stored in a database
  3. Data being read by an application

# Data At Rest

- So far we've been looking at encrypting Data In Motion.
- It's also important to keep certain data encrypted when at Rest.
- What kind of data might this be?

# Mobile Telecommunications

- Covering this topic because of the following in the Spec and on the past paper:
- It's also good to see how a non internet based communication system works

2.2 Describe the role of cryptography in a range of common public systems. For example, but not limited to:

- Mobile telecommunications

11 Voice privacy in Global System for Mobile Communication (GSM) cellular telephone protocol is provided by which cipher?

- A A5/2.
- B B5/4.
- C A6/2.
- D B5/8.



# Global System for Mobile (GSM)

- Most widely used Cellular communication protocol in the world.
  - How your phone communicates with the cell tower when making calls and sending texts.
- You probably know it as 2G, 3G, 4G, and soon 5G
- By far the most widespread Telecommunication system in the World.



# GSM Subsystems

- GSM is made up of several different **Subsystems**:
  - The Mobile Station (MS)
  - Base Station Subsystem (BSS)
  - Network and Switching Subsystem (NSS)
  - Operation Switching Subsystem (OSS)

# Mobile Station (MS)

- The Mobile Station is the handheld device owned by the user which they send messages with. In other words your mobile phone.
- From the GSM perspective the mobile station is comprised of:
  - **Mobile Equipment (ME)**
  - The hardware of the mobile phone, battery, antenna for sending/receiving data, microphone, speaker etc.
  - **The Subscriber Identity Module (SIM)**
  - Card which uniquely identifies the device on the mobile network.



# Base Station Subsystem (BSS)

- Is Comprised of:
- Base Transceiver Station (BTS)
  - Sends and receives signals to and from the MS, performs encryption, encoding, multiplexing etc.
- Base Station Controller (BSC)
  - Allocates Radio Channels to MS
  - Manages handover of MS to another
  - BTS



# GSM Diagram (Some context)

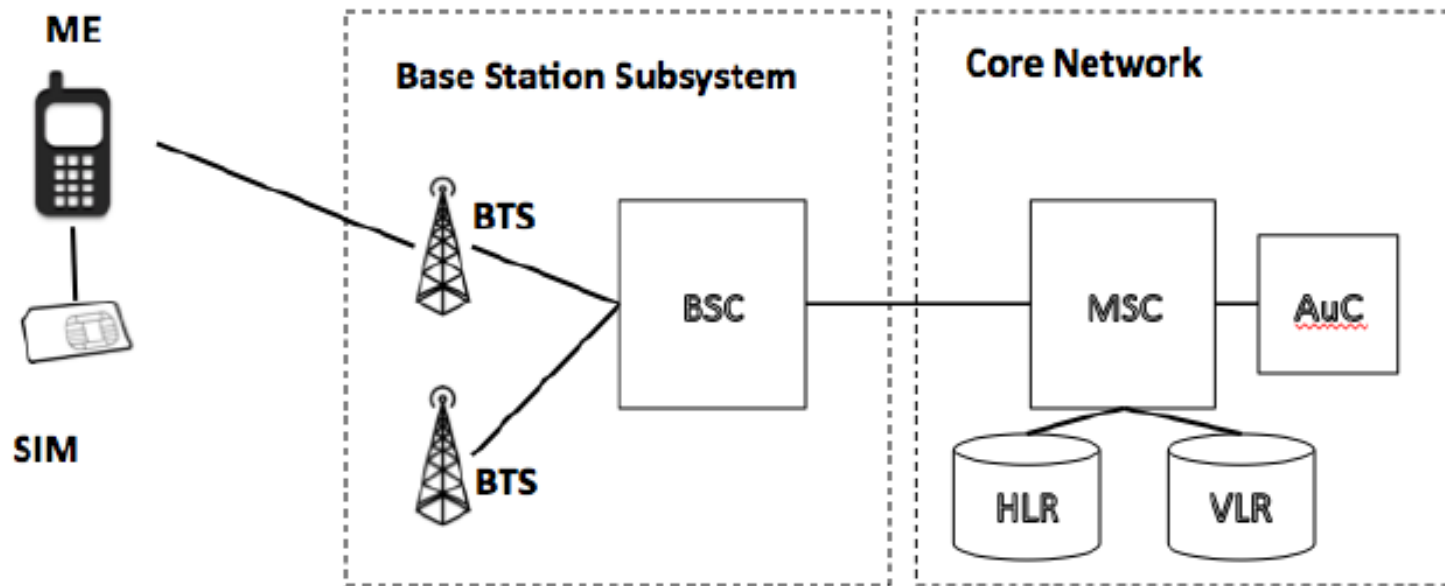


Diagram Credit:

[https://www.researchgate.net/figure/Structure-of-a-GSM-Network\\_fig1\\_262105075](https://www.researchgate.net/figure/Structure-of-a-GSM-Network_fig1_262105075)

# The Full GSM communication process

- <https://www.youtube.com/watch?v=1JZG9xVOwA>



# Extra Activity/ Homework

- Right now we don't have time to go through the:
  - Network and Switching Subsystem
  - Operating Switching Subsystem
- They're not the most interesting part of the system in terms of encryption.
- Learn about each of these and think about how the cryptographic systems they use might be different

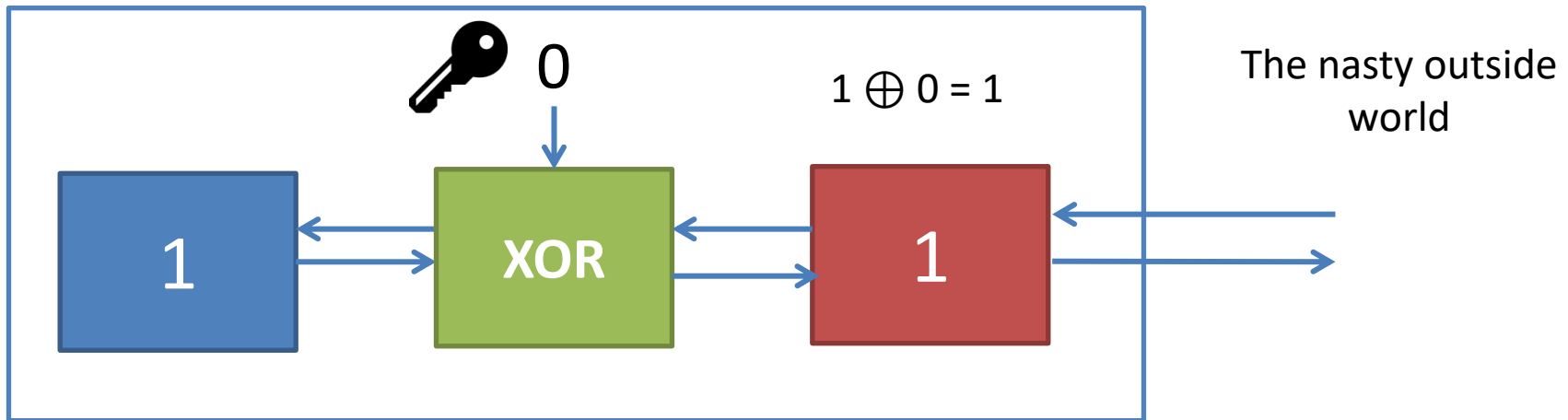
# GSM Security

- GSM allows 7 different stream ciphers to be used to Encrypt traffic between the Mobile Station and the Base Station Subsystem.
- By far the most common of these is called A5/2
  - A slightly more advanced version of A5/1
- The thing that differentiates stream ciphers from one another including A5/2 and A5/1 is the generation of the key stream.



# Stream Ciphers: Recap

- Stream ciphers XOR the key stream with the plain text to generate the Cipher Text.
- The Cipher text can then be XOR with the key to retrieve the plain text at the other end.
- Key is pseudo random and generated from a seed.



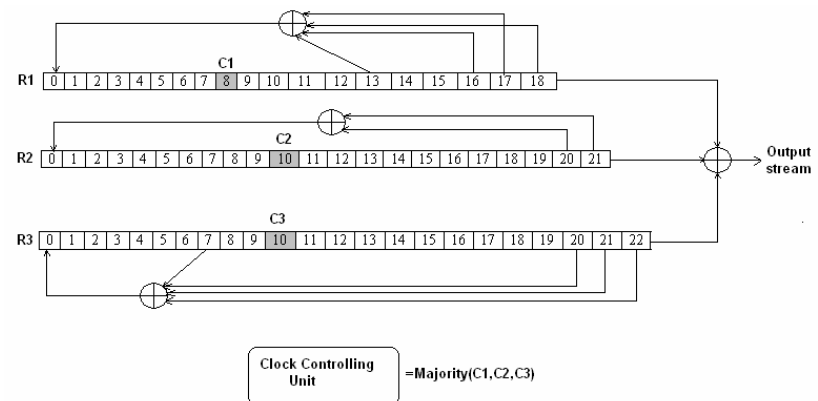
# A5/1 Key Generation:

## A Complete Stream Cipher example:

- The challenge with Stream ciphers is that you potentially need a really long key stream without patterns in it.
- The key stream needs to be generated from a small symmetric key which was transferred using an Asymmetric system  
(See 03AsymmetricCryptography)
- Think about the one time pad (See 02Introduction to Cryptography)
  - We don't want to send a key the same size as the message.
  - We want to send a small key but encrypt a large message effectively using that small key, without repetition in the cypher which could expose the key.

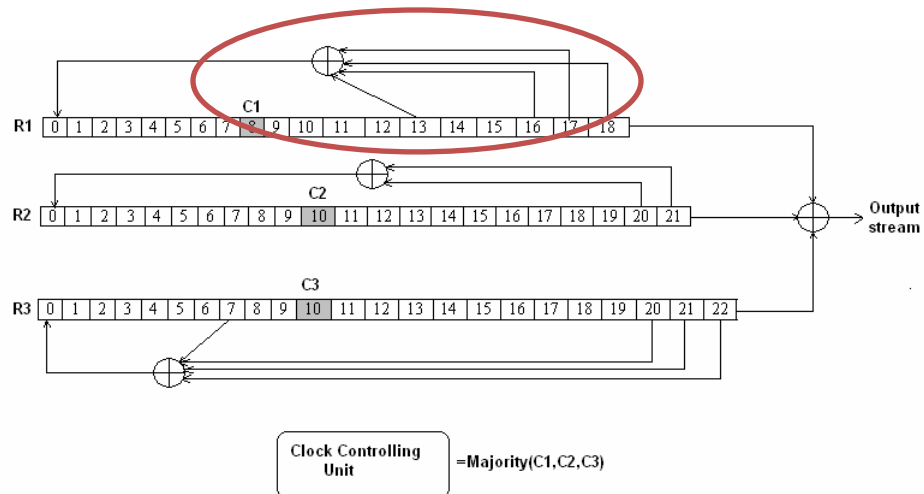
# Generating a Key Stream with A5/1

- A5/1 splits the 64 bit key into 3 blocks referred to as **Registers** these registers are 19, 22 and 23 bits long
- Each time we generate a bit in the key stream we start by looking at bits 9, 10 and 10 (highlighted in grey in the picture)
- The algorithm then says what is the majority value in those positions across the three registers (if there's two 0s and one 1 then the majority is 0, if there are two 1s and one 0 then the majority is 1)
- The two registers that form the majority perform a **Step** operation



# A5/1: The Step

- For each register which needs to perform a step we XOR the values in the positions with the arrow leaving them, circled in red.
- The first bit in the register is updated with the result of that XOR
- Finally XOR the bit in the last position of each of the three registers
- That Generates the value for the output stream



# A5/1 Main Takeaway.

- If both Alice and Bob have the 64 bit key and they both use this method they will generate the same key stream which can be made as long as necessary to encrypt and decrypt the data with.

