

# Remaining Topics

Jack Bradbrook

# Topics

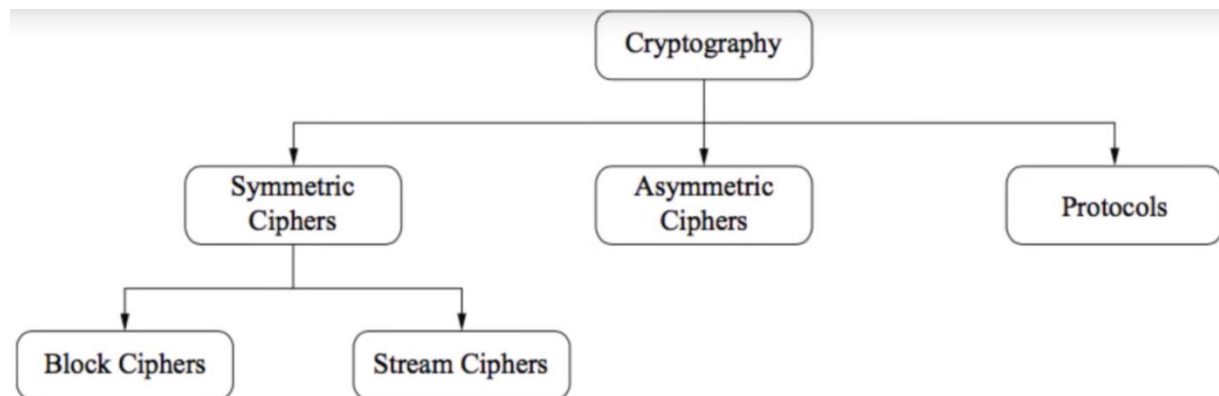
- Block Cipher Recap + Examples
  - How a block Cipher Works (Substitution and Permutation)
  - Example Block Ciphers (AES, DES)
- Terminology for ciphers
  - Monographic Cipher
  - Polygraphic Cipher
- Key Management
  - Key Rotation.
  - Key Expiry Date.
  - Kerberos key distribution.
  - Cryptographic key lifecycle.
- Attacks
  - Version Spoofing Attack
  - Reverse Protocol Attack
  - Version Rollback Attack
  - Downgrade Attack
  - Side Channel Attack
  - Password Attack

# Block Ciphers

## Some Examples + Recap

# Block Ciphers Recap

- Block Ciphers are a method of symmetric encryption, one key is used for both encryption and decryption.
- Block ciphers encrypt blocks of a message of a certain fixed size e.g. 64 bits.
  - If the message doesn't fit directly into a block padding is added (Extra values not part of the message) to fill the space.
- Block Ciphers are slower than stream ciphers but more secure



# Building the blocks

Dear Morgan, I am writing this confidential letter to you in hopes that we can arrange for a meeting. I know the sensitive nature of this subject will cause you some consternation. Do not worry. We will use encrypted messaging.

Convert to HEX

75 6d 20 64 6f 6c 6f 72 20 73 69 74 20 61 6d 65 74 2c 20 63 6f 6e 73 65 63 74 65 74  
75 72 20 61 64 69 70 69 73 63 69 6e 67 20 65 6c 69 74 2c 20 73 65 64 20 64 6f 20 65 69 75 73 6d 6f 64 20 74  
65 6d 70 6f 72 20 69 6e 63 69 64 69 64 75 6e 74 20 75 74 20 6c 61 62 6f 72 65 20 65 74 20 64 6f 6c 6f 72 65 20  
6d 61 67 6e 61 20 61 6c 69 71 75 61 2e 20 55 74 20 65 6e 69 6d 20 61 64 20 6d 69 6e 69 6d 20 76 65 6e 69 61  
6d 2c 20 71 75 6f 73 20 6e 6f 73 74 72 75 64 20 65 78 65 72 63 69 74 61 74 69 6f 6e 20 75 6c 6c 61 6d 63 6f

Split into Blocks

4c 6f 72 65

6d 20 69 70

73 75 6d 20

64 6f 6c 6f

72 20 73 69

74 20 61 6d

65 74 2c 20

# Data Encryption Standard (DES)

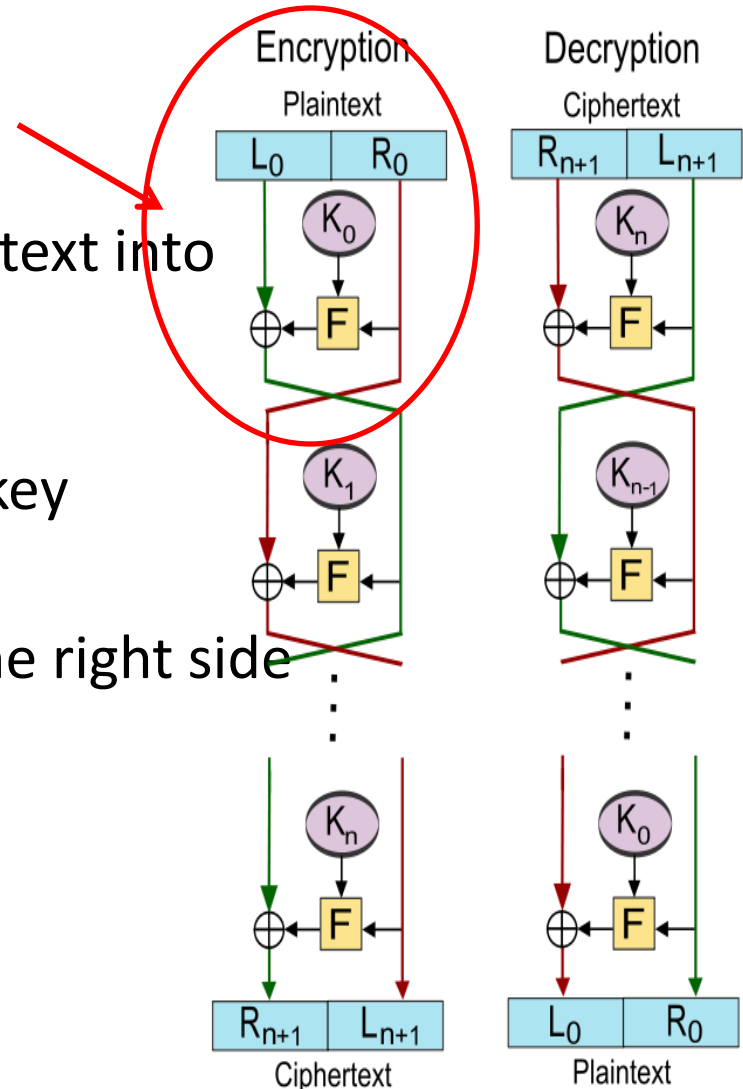
- Invented in the 1970s
- 56 bit key used to encrypt 64 bit blocks of data
- DES applies encryption using rounds, 16 rounds
- Each round has a 48 bit key built using the main 56 bit key to generate it, this is called **Key Expansion**
- To decrypt the message we reverse the process

# DES uses a Feistel Style Cipher

Method:

1. Split the Plain Text or the cipher text into left and right hand sides
2. XOR the left hand side with the key
3. Swap the XORed left side with the right side for the following round
4. Repeat this for multiple rounds

This is  
one  
round



# Advanced Encryption Standard (AES)

- Probably the most common block cipher used today.
- An Example of a Substitution Permutation Cipher.
- Has Superseded DES after it turned out DES had a back door that the NSA knew about, the keys for DES are also now considered to be too small given the speed of modern computers.
- Originally entered into a competition held by NIST (Remember them from the lecture on laws and Jurisdictions) to replace DES



# AES

- Encrypts blocks of 128 bits (16 bytes of data at once)
- If the message is not divisible by the block length then you add padding to fill out to the size of the next block

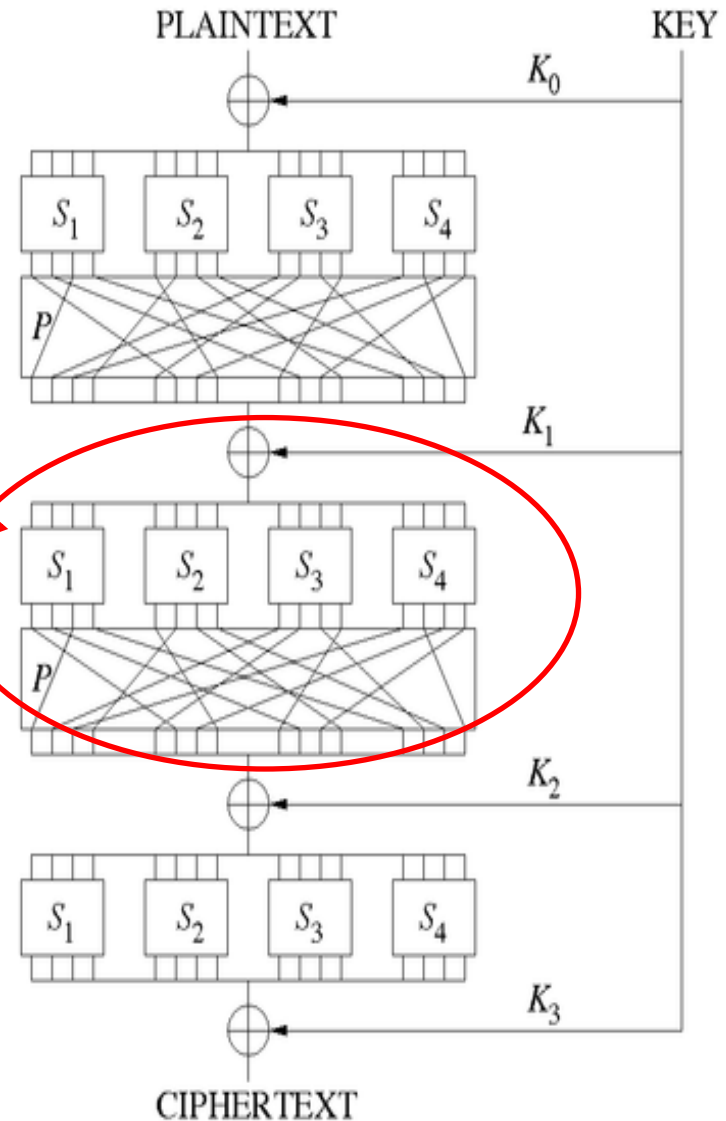
# AES Encryption Process

- As with DES AES uses Rounds to add encryption
- AES allows the user to select a security setting
- The security setting determines the number of rounds and the length of the key used.

Less Secure, Faster	Key Length	Number Of Rounds
↘	128	10
	192	12
More Secure, Slower →	256	14

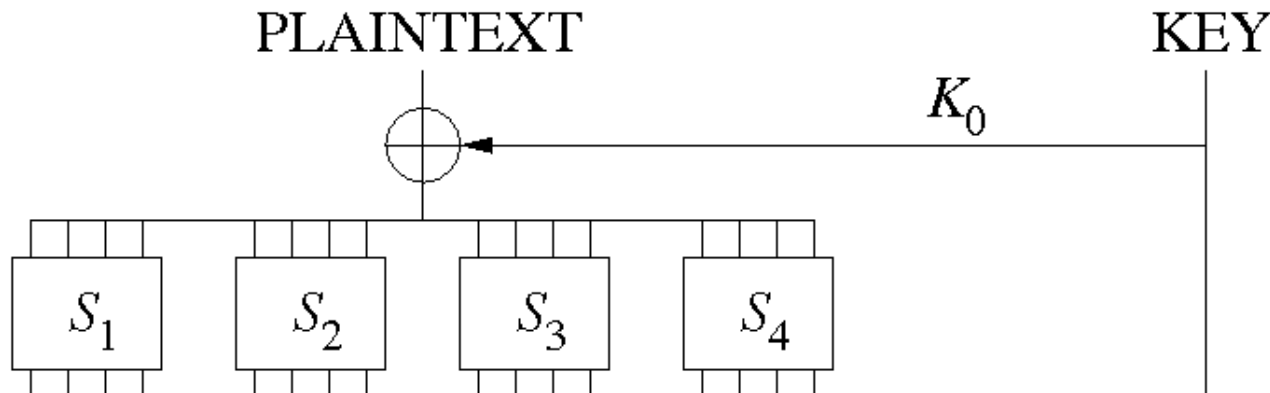
# Substitution Permutation (SP) Network

- Adds lots of Diffusion and Confusion!
- Hard to break with enough rounds.
- Going to explain principals of Substitution and permutation rather than exactly how AES does this, if you're interested in this you can research it.



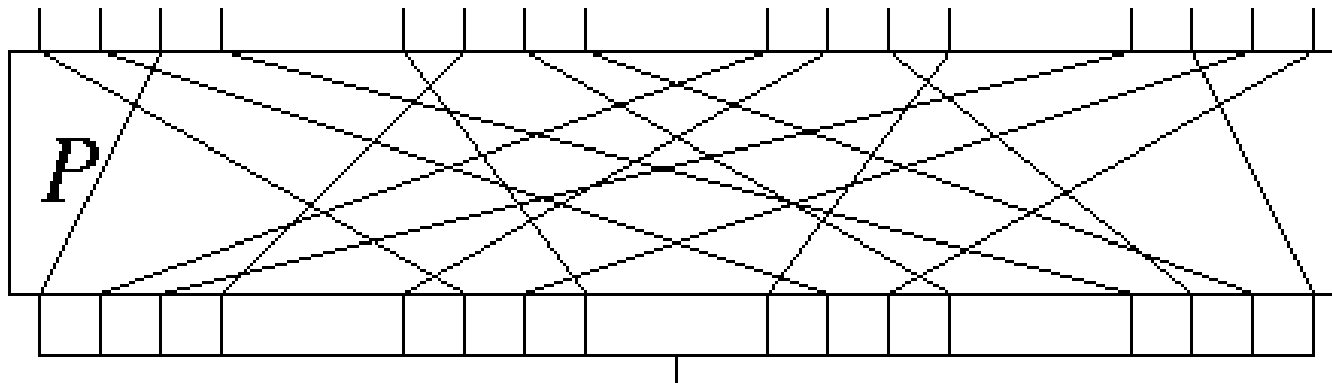
# Substitution Step

- This is the process that the old style ciphers: Caesar cipher, enigma machine all use, we map some value e.g. “a” onto another value e.g. “q”



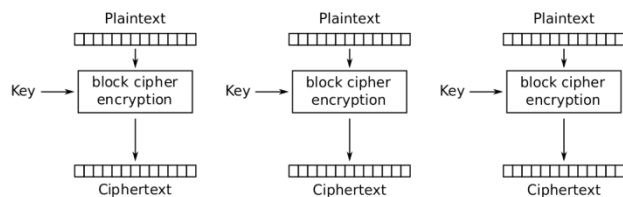
# Permutation

- Permutation is about moving parts of the message around without actually changing the values that make up the message
- E.g.  
“this is Jacks message” -> “is message Jacks this”  
Or “this is Jacks message” -> “kgasssii a cjesemths”

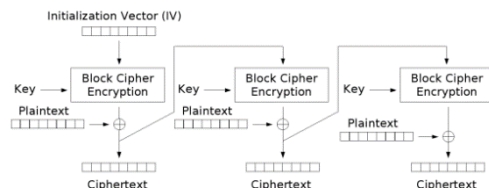


# Modes of Operations

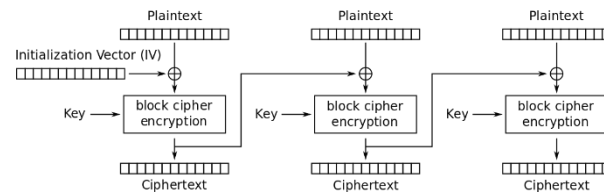
- DES and AES are flexible and can set up to use different modes of operation.
- Modes of operation exist in order to define the way encryption is applied across many blocks of data,
- The Fiesel and Substitution/ Permutation methods by contrast describe how encryption is applied directly to each block within the algorithm.
- For more detail on these see 04CryptographyInTheDigitalWorld



Electronic Codebook (ECB) mode encryption

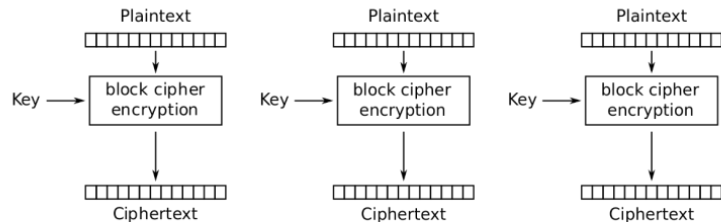


Cipher Feedback (CFB) mode encryption

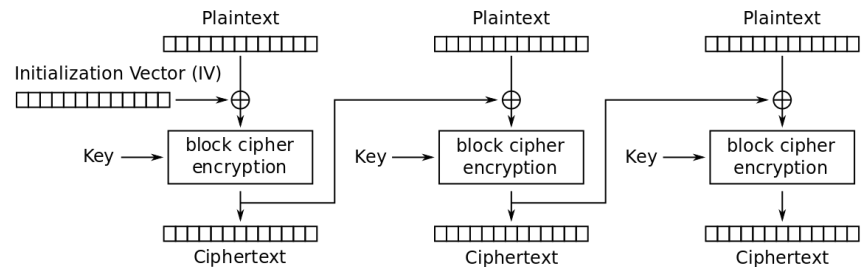


Cipher Block Chaining (CBC) mode encryption

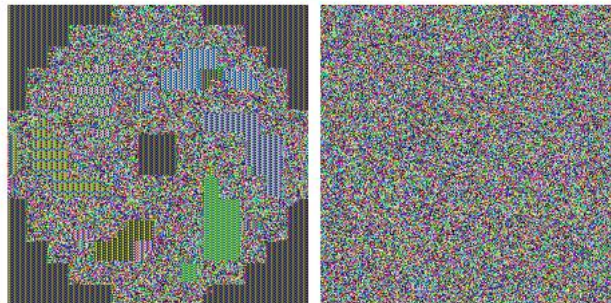
# A Final Quick Example



Electronic Codebook (ECB) mode encryption



Cipher Block Chaining (CBC) mode encryption



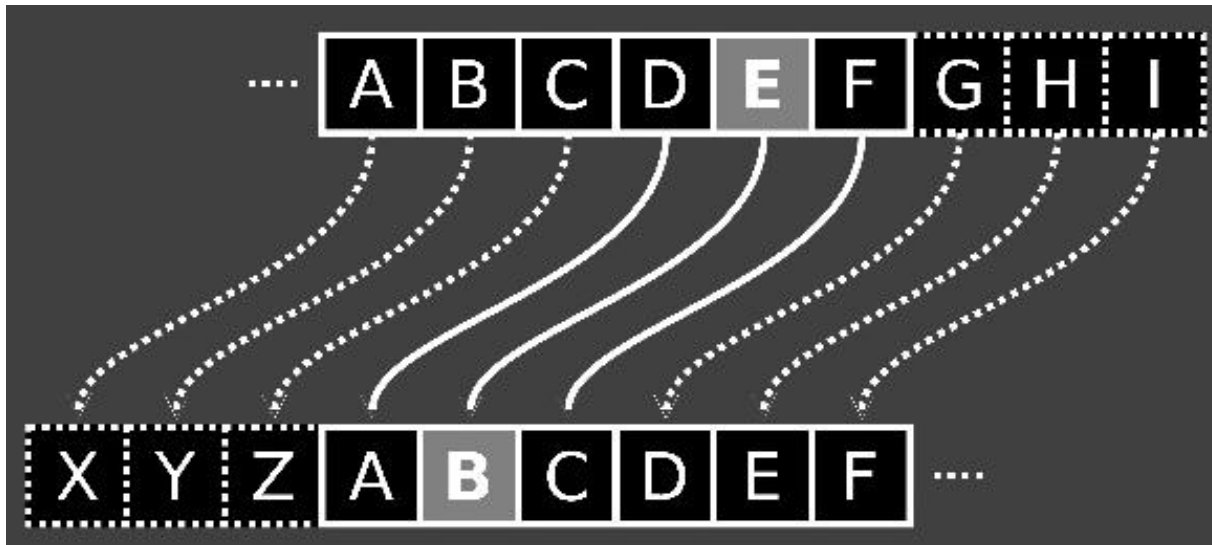
*The bitmap image encrypted using DES and the same secret key. The ECB mode was used for the left image and the more complicated CBC mode was used for the right image.*

# Substitution Cipher Types



# Monoalphabetic Cipher

- Each letter in plain text corresponds to one other letter in Cipher text.
- Caesar Cipher is a Monoalphabetic cipher



# Polyalphabetic cipher

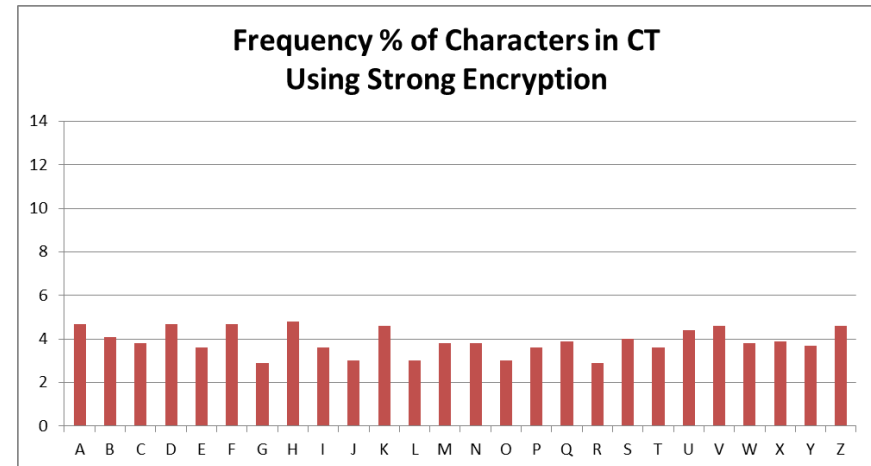
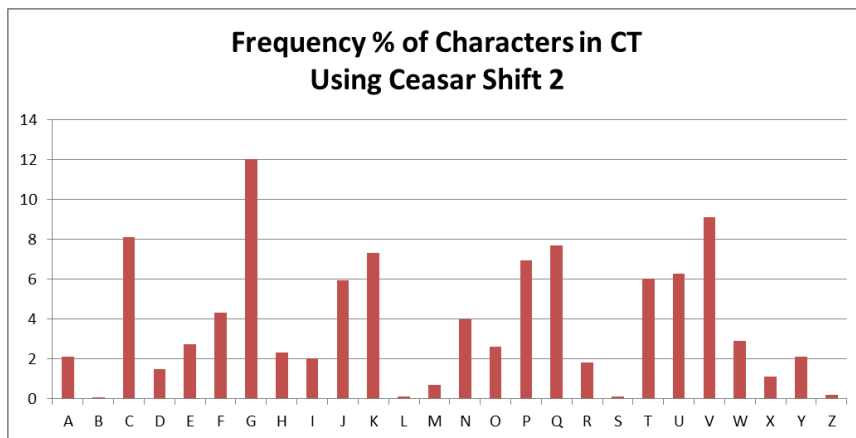
- Historical method used between the time of the Caesar cipher and systems such as Enigma
- With the Caesar Cipher we perform the same number of shifts for every character across the message.
- With a Polyalphabetic cipher we perform different shifts on each character based on the key.
- Take the key word and convert it into numbers based on letter positions in the alphabet
- E.g. "Jack" = 10-1-3-11

# Applying the Polyalphabetic Cipher

- Once we have converted our key like so:
- E.g. “jack” = 10-1-3-11
- We then loop the key shift values along the length of our message
- Message: “Thisisanexamplemessage”
- KeyLooping: “jackjackjackjackjackja”
- Shift values: 10-1-3-11-10-1-3-11-10-1-3-11-10-1-3-11-10-1-3-11-10-1
- CipherText: “Dijsstbnoybmzmfmoattaqf”
- The other party reverses this process using the same key.

# Polyalphabetic Cipher

- A Polyalphabetic Cipher will create a much more even frequency distribution.
- This makes it harder to break than the Caesar cipher.



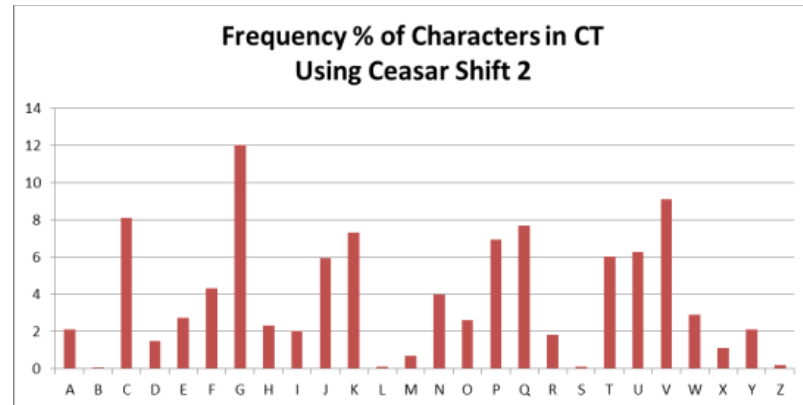
# Weakness of Polyalphabetic Cipher

- A Polyalphabetic Cipher is basically several Caesar ciphers at the same time.

- Suppose I split the message up into groups of the different shifts

- Thisisanexamplemessage ->

- Tiepeg
- Hsxlse
- laaes
- snmma



- Each of these sub groups of the string is simply a different Caesar cipher problem, there will be a clear frequency distribution for each of these.
- We can break the whole message by solving each of these groups and rebuilding the plain text message.

# Splitting the Groups Up

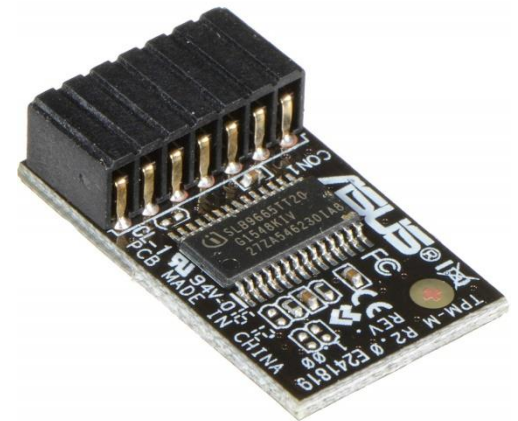
- The only remaining task is to split up the groups into multiple Caesar ciphers.
- To do this the length of the keyword needs to be determined.
- This is difficult but if you look at the letter frequency distribution for a large enough amount of encrypted data it's possible to do (It's much easier now with computers)

# Pseudo Random Number Generators in Use

- *Describe the use of dedicated hardware module for secure key and random number generation:*
  - *Trusted Platform Module (TPM);*
  - *Intel's Digital Random Number Generator (DRNG).*

# Trusted Platform Module

- A piece of hardware on a computers motherboard, typically a chip.
- The TPM Chip generates, stores and limits use of cryptographic keys
- Generates random numbers
- Can also perform hashing functions





# Why have TPM?

- Because TPMs are built as hardware they can be made tamper resistant.
- Software can be modified or deleted by someone with the right skills, hardware is physically part of the computer.
- Security is built into the architecture of the computer from the lowest layer possible.
- An attacker cannot therefore change those hardware methods for key generation random number generation or hashing to create a vulnerability they can exploit.

# Intel Digital Random Number Generator (DRNG)

- Bit of hardware on Intel Chipsets for generating random numbers
- Generates random values using measured fluctuations in the temperature to add entropy and make the values non deterministic.
  - Random values are based on the state of the outside world.
- Even if we understood the design of all the hardware and software for DRNG we still could not predict the values that it outputs.

# Key Management

# From The Spec:

- *Describe distribution practices:*
  - *Diffie-Hellman key exchanges;*
  - *Public Key Infrastructure;*
  - *Kerberos Key Distribution Centre.*
- *Describe key management best practices:*
- *unique key identification;*
- *key rotation;*
- *key expiration date.*

# Key Rotation

- If you have a small key and encrypt a very large amount of data using it you will have repetition.
- Even if the cryptographic algorithm used is very good, lots of confusion + diffusion, substitution, permutation if the key is used to encrypt a huge volume of data over lots of communication there will be reoccurring patterns which could be analysed and exploited by a skilled hacker.
- **Key Exhaustion** is using a key more times than it should be used.
- Because of this there needs to be a system in place for replacing the keys that we use when it's appropriate to do so.

# The Key lifecycle

- Cryptographic keys exist within a lifecycle
- The stages of this lifecycle are:
  - Generate – Generate the key
  - Distribute – Share the key with peers you will communicate with
  - Deploy – Start using the key to encrypt communication
  - Archive – remove the key from deployment
  - Destroy - Destroy the key so it cannot be used again

# Key Expiry Date

- It's up to organisations to have a plan as to what data certain keys are used to encrypt and for how long those keys are used.
- The longer a key is (the more bits it uses) the more secure it is. A longer key can be used for a longer duration before it is deemed insecure.
- When a key has reached the end of its life a new key needs to be generated, signed and then transferred so that can be used instead.

# Key Hierarchies

- Most organisations have a **Key Hierarchy** this determines what keys are used for what purposes.
- It's a good idea to encrypt different kinds of data with different keys.
  - More secure keys can be used for more sensitive data.
  - If a key is compromised it's easy to isolate the data encrypted with that key which could also have been potentially compromised



# Key Escrow

- Allow a second copy of private keys to be held by a third party.
- Allows that third party to access data that has been encrypted by you if you lose your key or you refuse to decrypt that information and provide it to them.

# Key Escrow

- Why would we want to do this though?
- Say you work for a company using a laptop with an encrypted hard drive, provided to you by the company.
- You fall out with the company and vow never to give them access to the files on the laptop when they retrieve it from you.

# Exercise:

- What's the obvious risk of a Key Escrow?

# The issue with this

- The company has the right to access the data on their own machine, your work contract should have specified that any work done on the machine was the property of that organisation.
- That company needs to use their copy of the key in order to decrypt the hard drive.

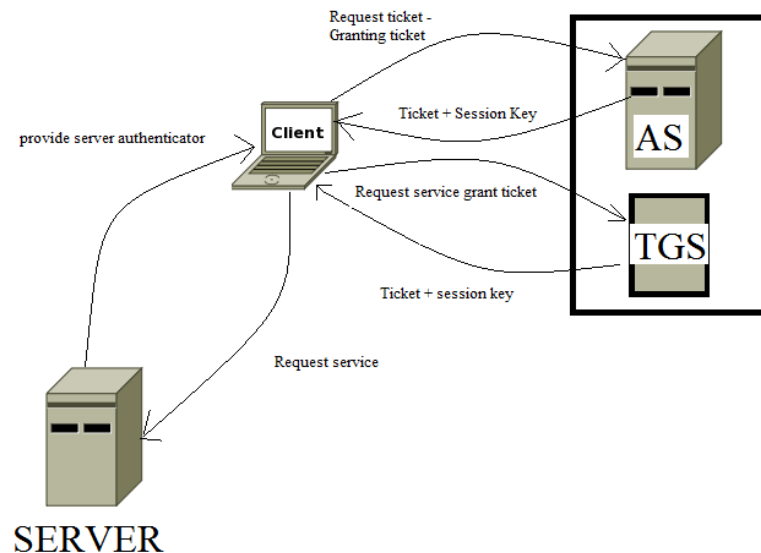
# Kerberos Key Management

# Authentication on a local network.

- Suppose you're within a local IT network that is secured from the outside world.
- As a member of that local network we're going to make use of a lot of resources available through the network
- Generating lots of public keys and applying lots of protocols to repeatedly provide authentication to lots of machines and get their symmetric keys using Diffie-Hellman is going to get really complex.
- What if you only had to prove your authenticity once before communicating with any of the machines on the network.

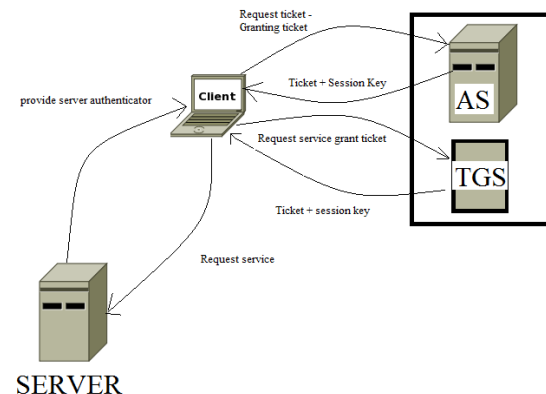
# Kerberos Key Management

- Kerberos is a system designed to allow somebody to authenticate themselves once and then communicate securely with many parties on a local network.



# Kerberos Overview:

- Using Kerberos a machine performs authenticates itself with a Key Distribution Centre (KDC)
- The KDC then grants the local user a Token which is valid for one day
- This Token can then be provided to other machines on the network as proof that the machine has been authenticated by the key distribution centre
- The other machines trust the KDC so trust the machine talking to them as well.
- There's a lot more detail regarding Kerberos but not going to go into it now.





# Attacks

# From the Spec

- *Explain construct of the most common attacks:*
  - *version spoofing attack;*
  - *reverse protocol attack;*
  - *version rollback attack.*
- *Consider attacks on passwords while designing a secure solution:*
  - *brute force;*
  - *dictionary;*
  - *rainbow table.*

*A candidate should be able to:*

- *Describe side channel attack.*
- *Describe downgrade attack.*
- *Explain the use of Salt Value in cryptography.*

# Spoofing Attack

- A more general purpose version of man in the middle attack.
- More of a human level attack than a machine level attack.
- Attacker pretends to be some person or service that they aren't.
- As the victim you may then provide them with information that you wouldn't hand out.

# Phishing

- A Phishing attack is a form of spoofing attack where when the attacker pretends to be a party you know and sends you a message claiming to need information such as your username and password or bank details.
- Often creates situation in which recipient of the email feels under pressure to provide the requested information.



Dear valued customer of TrustedBank,

We have recieved notice that you have recently attempted to withdraw the following amount from your checking account while in another country: \$135.25.

If this information is not correct, someone unknown may have access to your account. As a safety measure, please visit our website via the link below to verify your personal information:

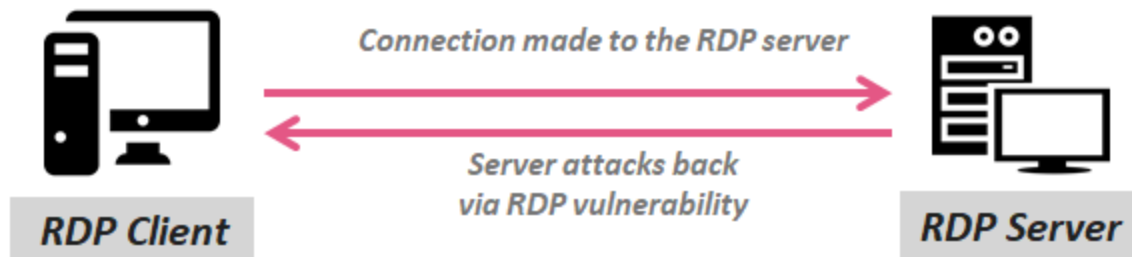
<http://www.trustedbank.com/general/custverifyinfo.asp>

Once you have done this, our fraud department will work to resolve this discrepency. We are happy you have chosen us to do business with.

Thank you,  
TrustedBank

# Reverse Protocol Attack

- A reverse protocol attack is when a protocol is modified so that when a client talks to a server the server responds with a message that causes the client machine to carry out something out which the client did not expect.
- If the attacker has control of the server they can modify its code to respond specifically with these messages.
- This attack works because in particular protocols a client application will perform actions based on the response it receives from a server without question. For example remote desktop applications.



# Version Rollback Attack

- Also called a “Downgrade attack”
- Occasionally a vulnerability in a security protocol is discovered
  - E.g. Somebody finds a way to get at data secured by people using that protocol
- When this is discovered the protocol is typically reviewed and patched by the body that controls it, once the patch is implemented users are encouraged to download it so they are no longer vulnerable
- What if you could persuade someone to still use the old version of a protocol though?

Version 3.0 → Version 2.0

# Version Rollback Attack

- Some Servers might say to the client “Hey I’m not ready to use protocol version 3.0 at the moment can we use 2.0 instead?”
- As the client the options then are
  - Use an older protocol version
  - Refuse to talk to the server
- Defence against this is often built in at an application level now, for example many web browsers will prevent users from accessing sites which require clients to use old versions of SSL or HTTPS.

Version 3.0 → Version 2.0

# Example: POODLE Downgrade Attack

- POODLE Stands for “Padding Oracle On Downgraded Legacy Encryption”
- Most well known downgrade attack
- Uses a man in the middle approach to gain trust.
- Makes users browsers resort to using old version of SSL protocol with known vulnerabilities instead of TLS.



# Exam Question

- Should now be obvious

**16** Which of the following is an example of 'downgrade attack'?

- A** MOODLE attack.
- B** POODLE attack.
- C** DOODLE attack.
- D** NODDLE attack.

# Other Downgrade Attacks

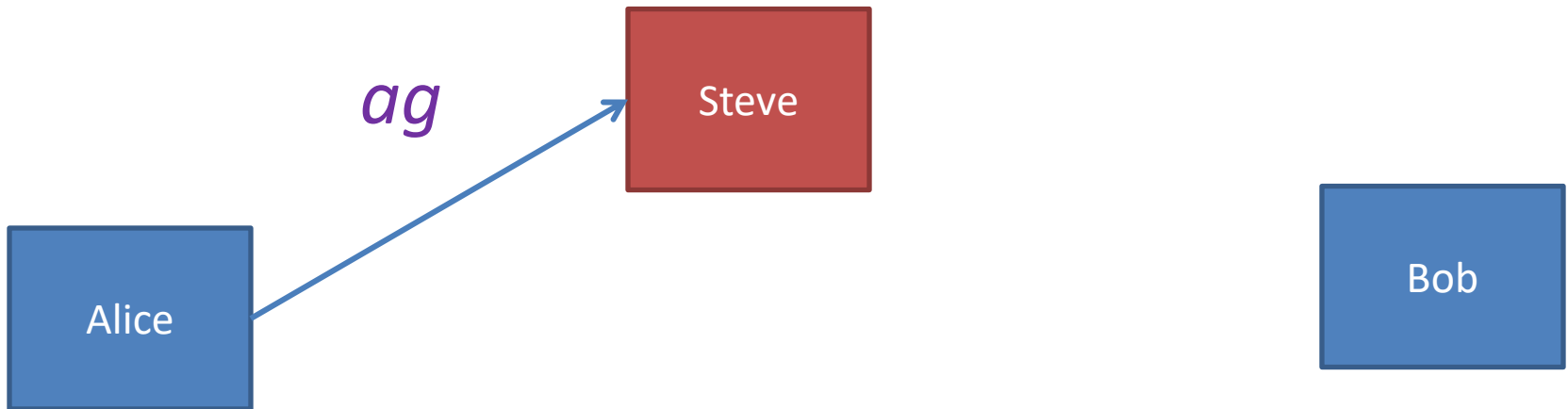
- POODLE is the most well known downgrade attack but some other famous ones are:
- Logjam (Man In the middle downgrade attack against Diffie Hellman, see 03AsymmetricCryptography)
- FREAK (Factoring RSA Export Keys)
- Just in case they ask you a basic question along the lines of “What is this an example of”

# Man In The Middle

- To perform a version rollback attack you first need to convince somebody that you are a trustworthy source.
- This is usually done by first performing some form of man in the middle attack whereby the attacker (Steve) intercepts communications between Alice and Bob and pretends to be each of them.
  - Alice thinks she is communicating with Bob but she is actually communicating with Steve.
  - Bob thinks he is communicating with Alice but he is also communicating with Steve.
- we covered this in 03AsymmetricCryptography but I will go over it again now

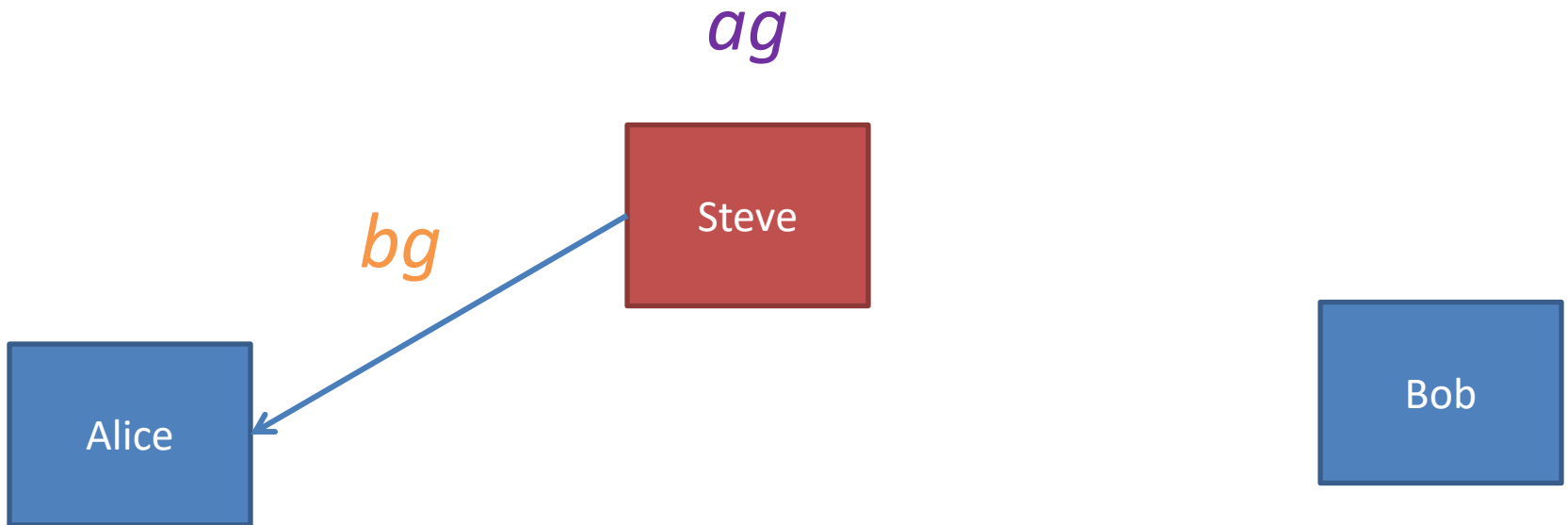
# Man In the middle example step 1

- Alice calculates *ag* and sends this to Bob
- The message is received by Steve who prevents the message from being forwarded to Bob



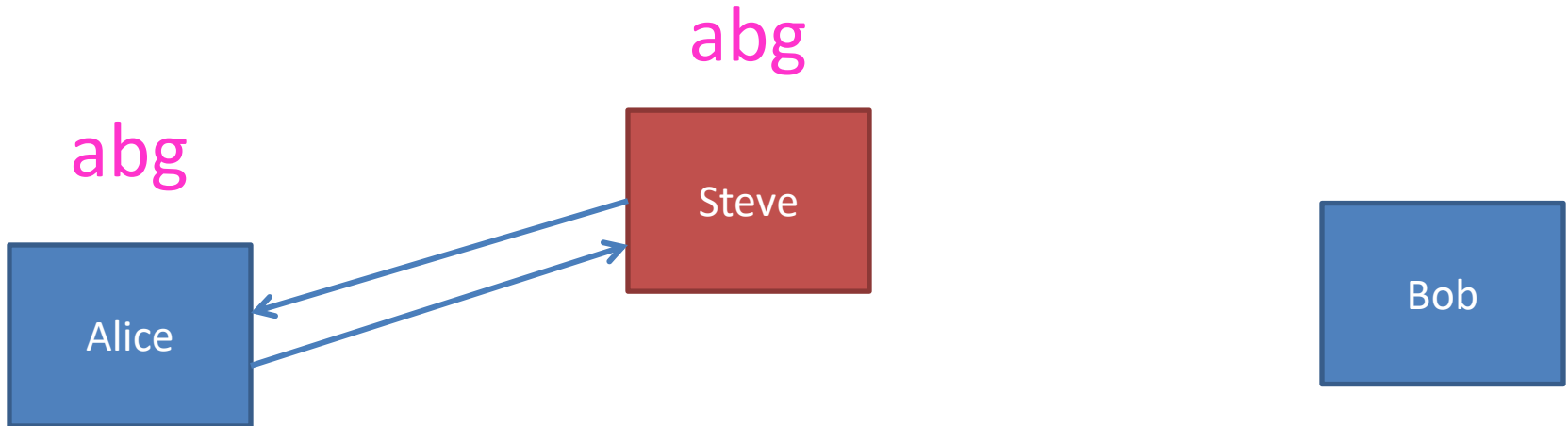
# Man In the middle example step 2

- Steve then Generates his own *bg* value as per normal and sends it back to Alice



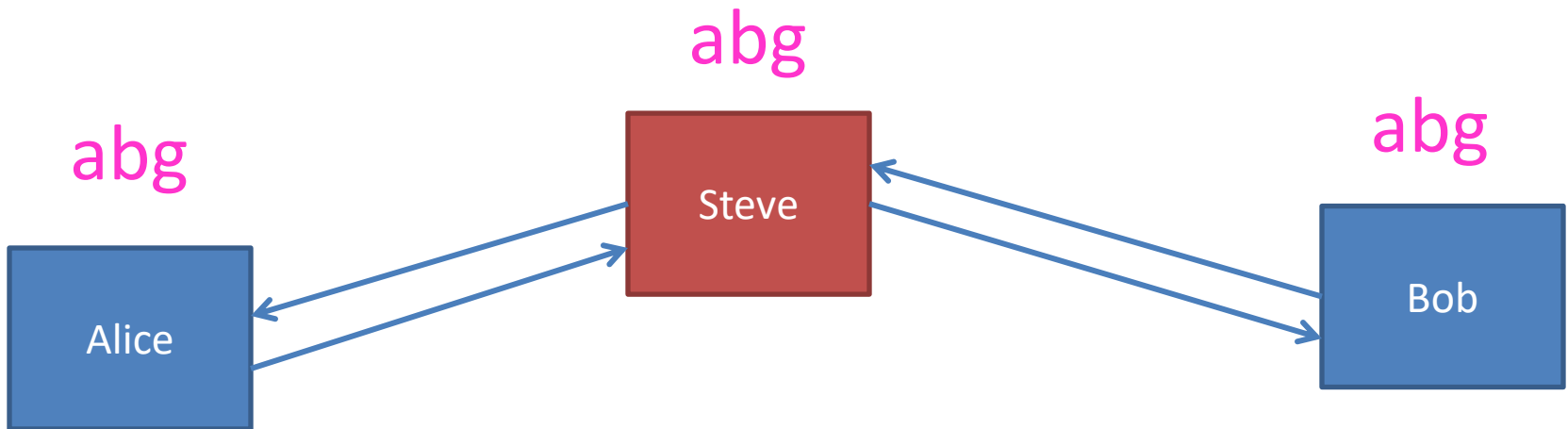
# Man In the middle example step 3

- Both Alice and Steve Complete the key exchange and arrive at an **abg** value.
- They can now both communicate with one another using a regular symmetric cipher.



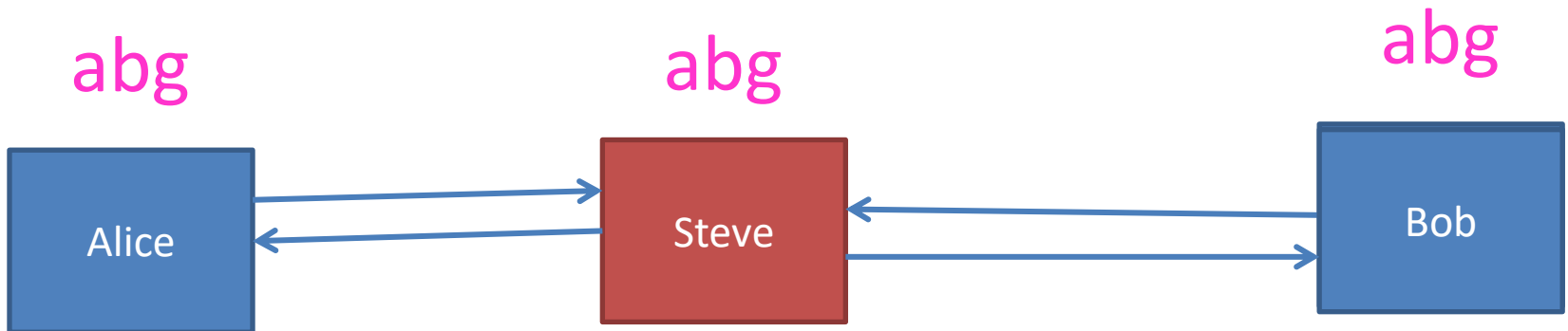
# Man In the middle example step 3

- Steve then repeats the process for bob to establish communication with him
- Steve then forwards messages between Alice and Bob but crucially he has control!



# Man In the middle example

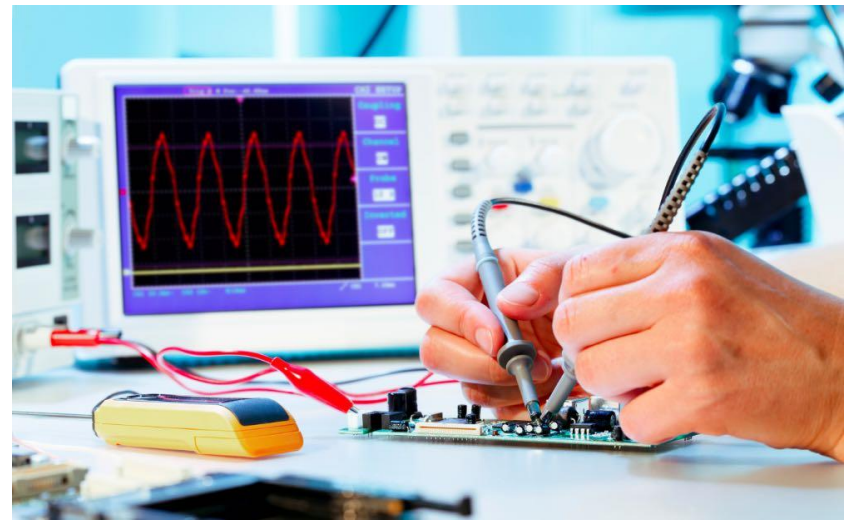
- Steve can now do the following:
  - Read all of Alice and Bobs messages to one another
  - Prevent messages from Alice or Bob from reaching the other
  - Write his own messages which he can send pretending to be the other party





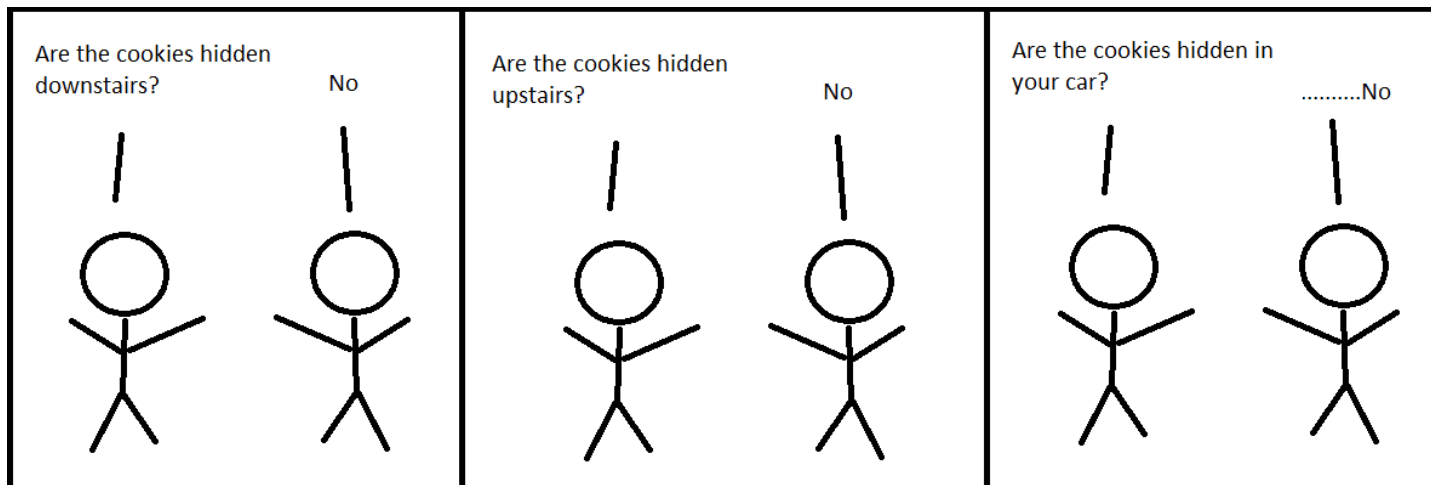
# Side Channel Attack

- Any attack based on an understanding of a weakness in the implementation of a system rather than the underlying algorithms used. To provide security.
  - E.g. exploiting flaws in the design of hardware that runs the algorithms



# Side Channel Specific Example: Timing Attack

- The timing attack is an example of a side channel attack.
- Suppose when a user logs into my system I have a method for checking if their password is correct.
- What might happen if my system takes different lengths of time to accept or reject the password?



# Timing Attack Explanation

- If we aren't careful about how we design our system a successful password which is nearly successful could take longer to check than an incorrect password.
- An attacker who realises this can design their password attempts to discover the correct password far more quickly

• One Guess: 

M	a	P	a	s	t	i	e
---	---	---	---	---	---	---	---

• Another Guess: 

M	Y	P	a	s	t	i	e
---	---	---	---	---	---	---	---

• Actual Password: 

M	Y	P	a	s	s	W	d
---	---	---	---	---	---	---	---

- The second guess might take longer for the system to return an "Incorrect Password" response to because the software will move along the password checking each character until all the characters are correct or it finds a character which doesn't match
- As an attacker we know we've got close to the correct password when the system takes longer to respond.
- We can use this to correctly work towards the correct password.

# Password Attacks

- If you want to uncover someone's password you have several options (Assuming a side channel style attack doesn't work):
  1. Obtain it by manipulation or social engineering e.g. trick them into telling you it by pretending you are a system administrator
  2. Use an algorithmic based attack to uncover the password

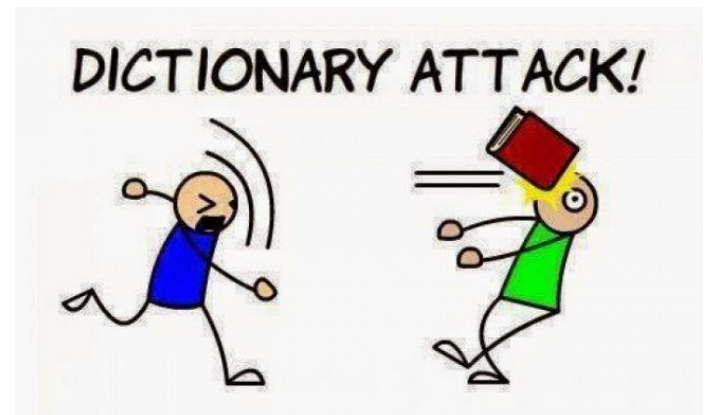


# Algorithmic Attacks on Passwords

- Most websites will allow a user to attempt to log just a couple of times before blocking attempts.
- Therefore a hacker typically needs to perform what is known as an **Offline Attack** whereby they gain a copy of the database containing the password hashes by infiltration of the system.
- Once the database has been obtained and the hacker knows the hashing algorithm that was used to hash the passwords they can make a large number of attempts to crack a password using software tools.
- All they have to do is input possible passwords into the hash function and see if it results in the hash value for a user in the database.

# Password Attack Hashing Strategies

- There are Three main methods used for breaking user passwords by “guessing”
- Brute Force Attack
- Dictionary Attack
- Rainbow Table Attack



# Brute Force Attack

- The brute force attack is the most basic method of guessing a password.
- Suppose we know the password is of length 5 and comprised of lowercase letters in the alphabet we would try every possible string which could exist up to length “aaaaa”, “aaaab”, “aaaac”...
- For that example you could have to try  $26^5 = 11,881,376$  passwords for each hash value in the database.
- We then hash each of these and compare these to a hash value in the database
  - Very slow
  - Extremely impractical for system where users have longer passwords with lots of characters in the password.
  - You have to do the same time consuming process for each hash value in the database.
  - It is 100% guaranteed get the correct password eventually however.

# The Dictionary Attack

- Most people choose passwords which are easy for them to remember rather than a random combination of letters.
- Their password is usually a name or word or some combination of words in the English language with some characters or symbols added or substituted for the letters.
- Therefore it's much better as an attacker to use a system whereby we check passwords in terms of the probability that they are the correct password.
- This is the purpose of The Dictionary Attack.
- The Dictionary Attack is effective against the majority of people who do not have complex passwords.



# Regularly Used Passwords

## What Are the 50 Most Common Passwords?

Based on most common duplicate passwords within a breach of over 30 million accounts.



SecurityScorecard

1. 123456	11. 123321	21. 222222	31. 333333	41. password1
2. 123456789	12. 1q2w3e4r5t	22. 112233	32. 123qwe	42. q1w2e3r4
3. qwerty	13. iloveyou	23. abc123	33. 159753	43. qqww1122
4. password	14. 1234	24. 999999	34. q1w2e3r4t5y6	44. sunshine
5. 1234567	15. 666666	25. 777777	35. 987654321	45. zxcvbnm
6. 12345678	16. 654321	26. qwerty123	36. 1q2w3e	46. 1qaz2wsx3edc
7. 12345	17. 555555	27. qwertyuiop	37. michael	47. liverpool
8. 1234567890	18. gfhjkm	28. 888888	38. lovely	48. monkey
9. 111111	19. 7777777	29. princess	39. 123	49. 1234qwer
10. 123123	20. 1q2w3e4r	30. 1qaz2wsx	40. qwe123	50. computer

# Dictionary Attack

- Make a list of likely passwords that we're going to check. This is the **Dictionary**.
- Move through the dictionary, Hash each password string and check if the hash value is the same as the hash of the user you want to break the password for.

# Building the Dictionary

- The main question left is: How do we build our dictionary so that we have a good chance of finding a password
- Most dictionaries used for password attacks are pretty big but are generally built using the same basic set of rules:
  - Start by adding common passwords (as I did in my example)
  - Then add each word in the English language
  - Add Common Names
  - Add Numbers and Symbols to the end of words e.g. “Jack12”
  - Perform common Character substitutions for parts of words and add those e.g. “Pa55w0rd”
  - Join different words already in the database together e.g. “JackPa55w0rd”
- We usually work through the dictionary in the same order as it is built when performing the attack.

# Limitations of the Dictionary Attack

- When performing a dictionary attack we're hashing each password in the dictionary and then comparing it to a specific password hash in the stolen database file.
- Hash functions are designed specifically not to perform the hashing operation super fast so that it's not possible to hash lots of values really quickly.
- We can't easily see if any of the passwords in the database are the same as any of the passwords we have hashed
- Ideally if we hashed a password in our Dictionary we'd like to see if any of the users who've hashed their passwords for the system used that password. Not just a specific user.

# Rainbow Tables

- Rainbow Tables are a method whereby we generate the hash for every string in our **Dictionary** and store it in the **Rainbow Table** before we perform the attack.
- We then simply go through the whole of the stolen database of hashed passwords and for each of the hashed passwords we do a lookup in the **Rainbow Table** to see if that hash exists in our Table.

# Pros and Cons of Rainbow Tables

- Rainbow Tables typically save lots of time if instead of caring about a specific account with a specific password.
  - We want to improve our chances of breaking into any of the accounts on the system (doesn't matter which one)
  - We want to break into lots of different accounts.
- Main Drawbacks
  - As an attacker it requires us to physically store a huge table so takes up a lot of space.
  - If all the users have chosen really poor passwords (passwords on top 10 worst password lists) we have wasted a lot of time and processing power pre computing the hundreds of thousands or millions of hashes for all the more obscure passwords in the table.

# Question?

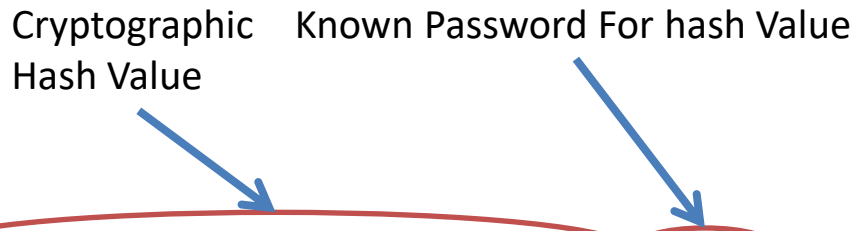
- Have we seen anything so far in the lectures which sounds like a Rainbow Table?

# Building the Rainbow Table

- Surprise! Rainbow Tables are just Hash Tables,
- We have a **key** and we can use this to access some data which is our **Value** really quickly.
- We can build a hash table whereby we look up a password based on the Cryptographic Hash value of that password, this is our **Rainbow Table**.

- It might look something like this:

Cryptographic Hash Value      Known Password For hash Value



The diagram shows a JSON object representing a hash table. Two blue arrows point from the labels 'Cryptographic Hash Value' and 'Known Password For hash Value' to the corresponding parts of a specific entry. The entry is circled in red. The hash value is '6367c48dd193d56ea7b0baad25b19455e529f5ee' and the password is 'abc123'.

```
{'7c4a8d09ca3762af61e59520943dc26494f8941b': '123456', '6367c48dd193d56ea7b0baad25b19455e529f5ee': 'abc123', '5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8': 'password', 'blb3773a03c0ed0176707a4f1574110075f7521e': 'qwerty'}
```

- We now have a system whereby we can perform the inverse of a cryptographic hash for specific hash values



# Rainbow Table Attack: Example In Python

```
# Import the code library which lets me do the hashing
import hashlib

# This is the Database of user password hashes which we stole somehow
stolenHashedPasswordDatabase = ["bc5351ffae3efe8067951f5deba4b294bf863f86", "b1b3773a05c0ed0176787a4f1574ff0075f7521e",
                                "333584fc2850d1a1f97a0a7bf8c5a12e684856bf", "26d58cf3df0903a2298788b72fced5bca9ea7144",
                                "7971ba9b9d4a9d20e2a323546018ba2833546053", "868657e908ac94e2b08ee7a83872dfc59d43f963"]

# Here we have just 4 possible passwords, in real life you would have hundreds of thousands of these.
reallyCommonPasswords = ["123456", "abcl23", "qwerty", "password"]

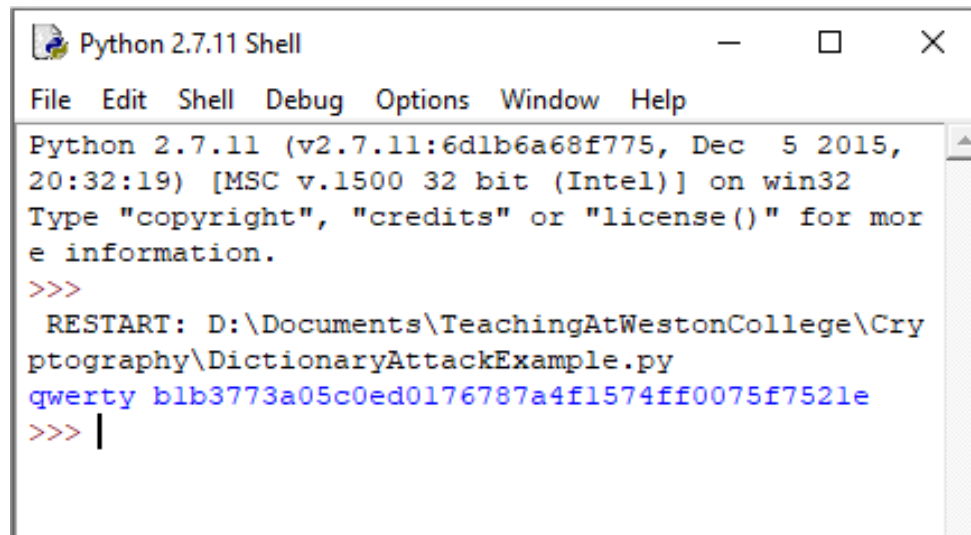
# The Dictioanry we're going to build and use to attack the stolen database with.
attackDictionary = {}

# Use the SHA1 algorithm to get a hash value for each of the possible passwords
# that we think someone might have chosen to use.
for password in reallyCommonPasswords:
    hash_object = hashlib.shal(password)
    hex_dig = hash_object.hexdigest()
    attackDictionary[hex_dig] = password

# Loop through all the stolen hash values and see if we have
# Any in the dictionary we created
for stolenHash in stolenHashedPasswordDatabase:
    # See if our dictionary has a lookup for the hash value.
    if stolenHash in attackDictionary:
        # If we found a hash for one of our possible passwords in the Database
        # Then print out the password and the hash value
        print(attackDictionary[stolenHash] + " " + stolenHash)
```

# Python Example Output

- You can check the answer using:  
<http://www.sha1-online.com/> for the Password “qwerty”



```
Python 2.7.11 Shell
File Edit Shell Debug Options Window Help
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015,
20:32:19) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for mor
e information.
>>>
  RESTART: D:\Documents\TeachingAtWestonCollege\Cry
ptography\DictionaryAttackExample.py
qwerty blb3773a05c0ed0176787a4f1574ff0075f7521e
>>> |
```

# A Solution to this Problem?

- If you were running a computer system where you stored the hashed values of peoples passwords
- What could you do to make the hashes resistant to a Rainbow Table attack?

# Password Salting

- A Salt is a Single Random Value generated for each password.
- It is added to the end of the password before the password is hashed each time.
- Each user has their own unique salt value associated with them.



# Salting

- Salt value = hash(password + salt)
- Suppose my password is Jack58 and my salt value is mk31p6
- The system would hash the value of Jack58mk31p6
- The hashed value of this would be stored in the database

# The Implication of salting

- As someone using a Rainbow Table attack salting is a problem.
- Salting means that almost all hash values associated with passwords stored in the database cannot be generated easily using a dictionary attack.
- The longer the salt string is the less likely an attack is to work e.g.
  - crHeD7jB8qkncG2q is a more secure salt than:
  - mk31p6

# What Next...

- That was the final PowerPoint of Material I have prepared.
- I will not be in College after today until next year.
  - This is the last time you will see me for this module.
- You have the test in several weeks after a few more revision sessions with Ian.
- I've presented the content and provided the learning resources for topics in the Spec
- You now just need to make sure you understand the content for the Test

# Further Preparation

- Please Make sure you go through the slides independently and think about each concept and how it applies to different, do some extra reading on each topic. I spent a lot of time on trying to make the slides as clear as possible so they can be used as a resource.
- I advise making your own set notes where you describe each concept in your own words, I've personally found this the best way of remembering information when preparing for tests in the past.
- I've designed the content to cover the topics listed in the specification in an ordering that makes sense. The slides are also designed to try to provide you with a good understanding of cryptography/ computer security in general. It's worth putting in the time to get to grips with the material, It forms the base for all computer security you will encounter.
- All the Powerpoints and python scripts will be stored in the module folder on Teams.
- If something isn't clear in the notes and you'd like some clarification do some of your own research or you can contact me at [jack.bradbrook@blueyonder.co.uk](mailto:jack.bradbrook@blueyonder.co.uk)
- In the test **make sure you read the questions carefully** and understand what they're asking, many of them are easy if you understand the question.