TECHNICAL SKILLS PRACTICE MASTER DOCUMENT

WEEK 5 CLASSICAL CRYPTOGRAPHY.
WEEK 6 ASYMMETRIC CRYPTOGRAPHY.
WEEK 7 CIPHER BLOCK MODES.
WEEK 8 HASHING.

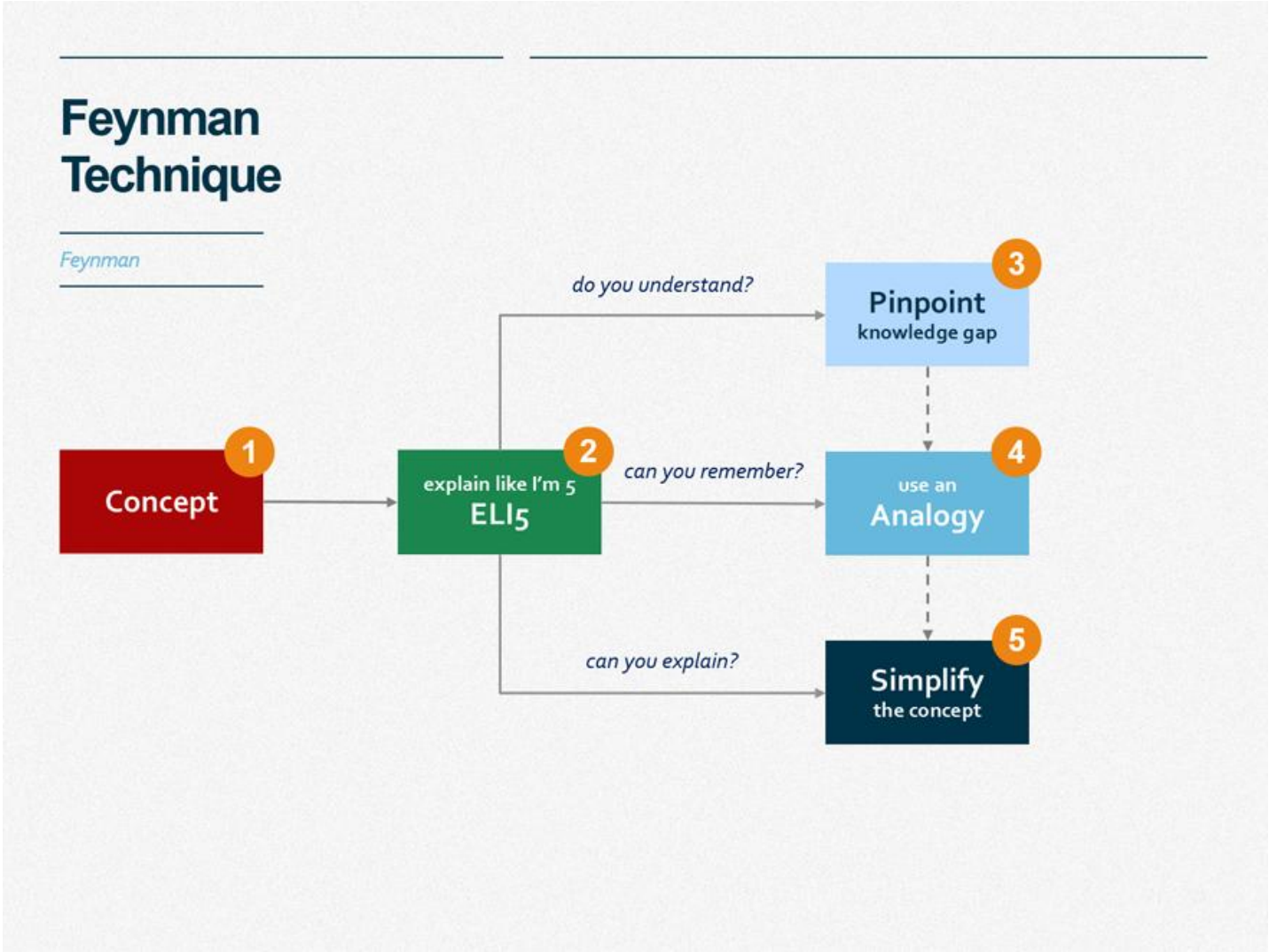
The ‘AnanISS' Guide to Technical Security: About.

This guide was compiled from material that was derived from student presentations in the ‘Annanas’ and ‘ISS’ tutorials of COMP6447: Security Engineering held at UNSW in Semester 1 2017.

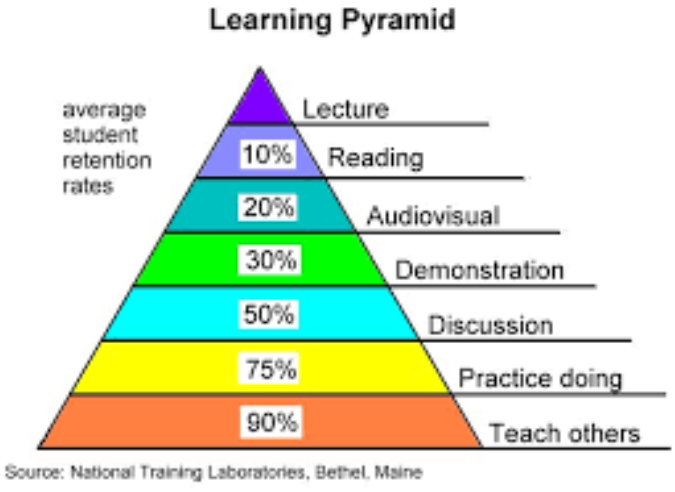
All of the material contained herein is understood to be open source and available to anyone to freely use, however, care should be taken in reference to some of the correctness and validity of the solutions as they have not been extensively error checked or professionally edited.

It should be noted that this guide covers only a subset of the examples for each relevant topic taught in COMP6447 and care should be taken when revising for the final exam that all examinable topics are considered in your study plans.

The Feynman Technique: A Simple Mental Model for Revising.



The Learning Pyramid: Student retention rates.



TECHNICAL SKILLS PRACTICE

WEEK 5 CLASSICAL CRYPTOGRAPHY

ROT - N.
VIGENERE CIPHER.
ONE TIME PAD.
CRIB DRAGGING.

INTRODUCTION: CLASSICAL CRYPTOGRAPHY.

Classical Cryptography refers to ciphers that were historically important and significantly used throughout history but have now largely fallen into disuse when compared to modern ‘Strong ‘ cryptography (with the exception of the One-time Pad), Classical Cryptography often relies on the secrecy of the keys to ensure confidentiality (in violation of Kerckhoff’s principal) this contrasts with modern cryptography which is based on hard mathematical problems such as prime factorisation in Number theory, which are clearly understood but hard to break and reverse.

Most classical ciphers are considered very easy to break using simple frequency analysis, computation and even hand calculations, however, the exception to the rule in the classical world is the One-time pad, the One Time Pad, if correctly implemented is considered the only unbreakable encryption technique.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

FIG1: A TABULA RECTA*

ROT N

ROT-N is a simple mono-alphabetic substitution cipher where letters in the plaintext are replaced by their counterparts in a version of the same alphabet which has been rotated by 'n' places. 'Rotating' involves assigning each letter a number 'n', which corresponds to a second letter, 'n' spaces down the alphabet. This rotation then gives an illegible through easily reversible letter output in the ciphertext for each letter in the plaintext.

Encryption of a letter *x* by a shift *n* can be described mathematically as,
$E_n(x) = (x+n) \bmod 26$

Decryption is performed similarly,
$D_n(x) = (x-n) \bmod 26$

ROT-N encrypted ciphertexts are easily decrypted using either brute force methods or frequency analysis.

EXAMPLE: An ROT-13 ENCRYPTION.
Consider the string s : s = 'SECURITY ENGINEERING'
$E_{13}(s)$ = 'FRPHEVGL RATVARREVAT'
FUN FACT: for ROT-13 the exact same process can be used to encipher and decipher, ie $E_{13}(x)=D_{13}(x)$.

SUMMARY OF IMPORTANT FEATURES.

- Rotation cipher: Characters are shifted by N places in same alphabet to encipher them.
- Cipher-texts can be brute forced for all N.
- Easy to analyze frequency and find patterns to discover the rotations and discover the message.

VIGENERE CIPHER

The plaintext's spaces are removed, and 'words' are attained by breaking the string into sections at every 'n' places. Much like in a Running Key cipher, this altered plaintext is aligned with a repeating key (a word of equal length to n). The two aligned letters are entered into a Tabula Recta (**See intro page***) and the extrapolated letter is used as the cypher text.

EXAMPLE: VIGENERE ENCRYPTION.

SECUR ITYEN GINEE RING ← Plaintext.
MELON MELON MELON MELO ← Repeated Key.
EINOE UXJSA SMYSR DMAU ← Ciphertext.

As can be seen the first cipher text character corresponds to the position $T_{S,M}$ using matrix notation for the Tabula recta matrix T, the rest of the characters follow the same encryption by lookup in the tabula recta.

Decryption simply reverses this process and is performed by going to the row in the table corresponding to the key, finding the position of the cipher-text letter in this row, and then using the column's label as the plaintext.

i.e: In our example for the first cipher text character E, going along row M from the key to position E from the cipher text corresponds to column label S ie the plaintext.

SUMMARY OF IMPORTANT FEATURES.

- Uses a repeated fixed length key for encryption.
- Susceptible to cryptanalysis due to repetition in the key.

ONE TIME PAD.

The One-time Pad (OTP) is a cipher that is completely uncrackable if implemented correctly. The cipher works by assigning a truly random key of at least the same length as the message to two parties, The plain text is combined with the truely random key in some way, There are a multitude of ways in which the ciphertext letters can be generated; for example modular techniques can be utilised to generate ciphertext letters by addition (mod 26), digits in addition (mod 10) or binary strings in addition (mod 2), In computers encryption is often achieved by performing a exclusive or (XOR, symbol ⊕) of a plaintext against a random key.

EXAMPLE: OTP ENCRYPTION IN ADDITION MOD 26

S	E	C	U	R	I	T	Y		E	N	G	I	N	E	E	R	I	N	G	**Plaintext**
19	5	3	21	18	9	20	25		5	14	7	9	14	5	5	18	9	14	7	**Alphabet #**
12	10	13	21	1	26	8	20		22	7	17	17	16	24	7	22	26	10	18	**Key**
31	15	16	42	19	35	28	45		27	21	24	26	30	29	12	40	35	24	25	**Sum**
5	15	16	16	19	9	2	19		1	21	24	0	4	3	12	14	9	24	25	**Mod 26**
E	O	P	P	S	I	B	S		A	U	X	Z	D	C	L	N	I	X	Y	**Ciphertext**

SUMMARY OF IMPORTANT FEATURES.

- A theoretically perfect approach to encrypt a fixed length message using the same length truly random key.
- Symmetry of the keys and difficulty generating truely random string limit its application in the real world.
- Crib dragging can be used when keys are reused.

CRIB DRAGGING

Crib-dragging is a method of performing a cryptanalysis against an incorrectly used implementation of a One-time pad. That is, it's a way of performing cryptanalysis when multi-time pads are used (ie. the same key is used for different messages). One-time pads are usually implemented electronically by taking a XOR of a plaintext message against a random key. The reason being the XOR operator has roughly a 50% chance of outputting a 0 or 1 This is in contrast to the OR (v) operation that has a 25% chance of outputting a 0, or the AND ($\wedge$) operation which has a 75% chance of outputting 0.

Breaking a Multi-Time Pad using Crib dragging.

Following these steps:

- Guess a word or phrase you think may occur in one of the messages, sensible to start with a common word like 'the' or 'is'.
- Encode the word or phrase from the above step.
- XOR the two ciphertexts together.
- XOR the string from step two at each position from step three.
- If the result is unreadable, move along positions and expand the crib search.

When the guess and the two XOR'd strings are lined up correctly, if the guess matches a word in the first message, the output will be the aligned word in the second message; and visa versa.

TECHNICAL SKILLS PRACTICE

WEEK 6 ASSYMETRIC CRYPTOGRAPHY

BITS OF SECURITY.
MERKLE PUZZLES.
DIFFIE HELLMAN KEY EXCHANGE.
RSA.

INTRODUCTION: ASSYMETRIC CRYPTOGRAPHY.

Asymmetric or Public-key cryptography is any crypto system that generates pairs of keys, these are usually split into two types of keys, **public keys** that are disseminated widely and can be used to encrypt a message and **private keys** known only to the owner that can be used to read a message when encrypted with the public key.

In this way Asymmetric cryptography functions like a PO box at a post office, anyone is free to send a message to the PO box (analogous to encrypting a message with the public key) but only the owner of the PO box can read the messages (Decrypt them using the private key).

Some notation for Public-key cryptography.

There is a set of encryption functions E_k with encryption keys k, there is a corresponding set of decryption functions D_{k^-} with decryption functions k^- such that,

1) For all k, there exists a k^- such for all m, $D_{k^-}(E_k(m)) = m$
2) It is easy to generate a random k and its paired k^- .
3) $E_k(m)=c$ is easy to generate, $D_{k^-}(c)=m$ is easy to calculate.
4) without knowing k^-, $E_k(m) = c$ is hard to invert.

Background knowledge: Trapdoor Functions

A trapdoor function or a one-way function f is one for which,

1) given x it is easy to find f(x).
2) given f(x) and nothing else, it is hard to find x.
3) given f(x) and some small extra information, it is easy to find x.

The prototypical trapdoor function is the discrete logarithm problem, which is given a large prime p and primitive element g in $\mathbb{Z}_p$, finding $f(x) \equiv g^x \pmod p$ is easy to calculate, however finding x given just g^x is called the **Discrete logarithm problem** and is computationally intensive.

BITS OF SECURITY

In cryptography, **Bits of security** is a measurement of the number of operations needed to discover the key of a cryptographic primitive (such as a cipher or hash) in a brute force attack, as measured in binary bits. In the worst case number limited by the key' s range of possible values (called its **keyspace**).

For a binary key of length n, there are 2^n possible values in its key space so we say the key offers n bits of security, however on average we would expect an attacker to discover the key after trying half the number of operations in the key space or 2^{n-1} operations.

EXAMPLE: BITS OF SECURITY.

A given 3 GHz, quad core PC can do 2 operations per clock cycle, given this machine, calculate how long on average (hours/days/years etc) it would take to brute force an encryption algorithm with a key that has 128 bits of security?. Helpful Relations: Speed (operations/sec) = Clock (cycles/sec) × Cores × operations/cycle.
$2^{10} \sim= 10^3$.

Machine Speed = 3×10^9(cycles/second) $\times 4 \times 2$(operations/cycle)
$= 3 \times (10^3)^3$(cycles/second) $\times 2^2 \times 2$(operations/cycle)
$\sim= 2^2 \times 2^{30}$(cycles/second) $\times 2^2 \times 2$(operations/cycle)
$\sim= 2^{35}$ (operations/second)

We Can expect to brute force a 128 bit key after 2^{127} operations on average, this would take $(2^{127} / 2^{35}) = 2^{92}$ seconds $\sim= 1.5 \times 10^{20}$ years!.

SUMMARY OF IMPORTANT FEATURES.

- Bits of security measures the computational complexity of trying to break a cryptographic algorithms key by brute force.
- For a binary string of length n worst case attack is 2^n operations, average case we can expect 2^{n-1} operations.

MERKLE PUZZLES

Merkle's puzzles were an early attempt at establishing a protocol for a public key cryptosystem, these days the protocol is largely used for pedagogical purposes rather than being practically implemented. Succinctly Merkle's puzzle's is the protocol that two parties can exchange a key by having one participant generate a large number puzzles such that each puzzle is easily brute forcible and contains a unique identifier and a key which can be stored and looked up later, the generator of the puzzles exchanges all of the puzzles with the other participant who solves just one puzzle and sends back the identifier, the generator of the puzzles looks up the key corresponding with the identifier sent back from the other party and they now have a shared key to communicate with, an attacker trying to discover the key would have to solve on average half of the puzzles and thus there is a huge discrepancy in time complexity of key generation vs a brute force attack on finding a particular key.

EXAMPLE: MERKLE PUZZLES.

Alice wishes to establish a shared secret with Bob over a 'leaky' channel (ie Eve is listening!), to do this she decides to use a system of Merkle Puzzles to establish the shared secret, if she uses 10 000 puzzles that are individually crackable in 2 hours for her implementation show the work discrepancy between Bob and Eve.

Assuming Bob picking a puzzle is negligible to crack a puzzles takes him only two hours of work.

Contrasting this Eve must solve on average half of Alice's puzzles or 5000 puzzles, if they take 2 hours to solve, it will take her on average 5000 X 2 hours =10 000 Hours to find the key Bob and Alice are using to communicate with.

SUMMARY OF IMPORTANT FEATURES

- For m puzzles crackable in n time, key generation is O(m+n). The attacker needs to do O((mn)/2) work on average.

DIFFIE HELMAN KEY EXCHANGE

Diffie-Hellman key exchange is a cryptographic algorithm for exchanging keys over a public channel, it operates in a similar way to Merkle puzzles in that the security of the keys is protected by an understood but computationally intensive problem, in this case, security is maintained by the Discrete logarithm problem.

EXAMPLE: ALGEBRAIC DESCRIPTION OF DIFFIE-HELLMAN PROTOCOL.

A and B wish to exchange a common key, with people listening in, in such a way that at the end the listeners do **not** know the common key.

Steps in Diffie-Hellman key exchange,

1) A chooses a large prime P used for modulus and a generator g and sends p and g to B (NB: a generator g is a primitive element of the group $\mathbb{Z}_p$, though this is not essential to know).

2) A chooses a secret number a.
   B chooses a secret number b.

3) A computes $x = g^a \pmod p$ and sends it to B.
   B computes $y = g^b \pmod p$ and sends it to A.

4) A calculates $y^a = g^{ab} = k \pmod p$.
   B calculates $x^b = g^{ab} = k \pmod p$.

5) A and B now communicate in secret using the shared secret key k.

SUMMARY OF IMPORTANT FEATURES

The Diffie-Hellman protocol allows two parties to exchange a common key over a public channel, in practice the protocol is combined with some form of authentication as man in the middle attacks are possible on the basic implementation.

RSA

RSA is a quintessential example of a public-key cryptosystem, the algorithm generates a public encryption function with a corresponding private decryption function, hence it eliminates the need to privately exchange keys via a secure channel.

RSA has some variations in implementation, however, the following implementation is simple and prototypical and uses **Euler's totient function** denoted by $\phi(n)$, the function $\phi(n)$ counts the number of positive integers up to a given integer n, that are relatively prime to n. For primes p,q we have,
$\phi(p)=p-1$, for any prime p.
$\phi(pq)= \phi(p)\phi(q)$, ie function is multiplicative if p,q co-prime.

EXAMPLE: ALGEBRAIC DESCRIPTION OF RSA CALCULATIONS USING EULER'S TOTIENT FUNCTION.
Each user does the following,

1) Chose 2 random large primes p,q (100 or 200 digits), find $n=p*q$ and calculate $\phi(n) = (p-1)*(q-1)$, where $\phi(n)$ is Euler's totient function.

2) Chose a random number e such that $1<e< \phi(n)$ for which $\gcd(e, \phi(n))=1$ (ie e and $\phi(n)$ are co-prime) and find
$d \equiv e^{-1} \pmod{\phi(n)}$, this exists since $\gcd(e, \phi(n)) =1$.

3) The public key is the pair (n,e) = k.
The private key is the pair (n,d) =k^{-}.

The encryption algorithm is (for $0<= m<n$)

$E_k(m) \equiv m^e \pmod n$

and decryption is achieved by

$D_{k^{-}}(c) \equiv c^d \pmod n$.
$\equiv m$. (****RSA has Technical restrictions, see Appendix).**

TECHNICAL SKILLS PRACTICE

WEEK 7 CIPHER BLOCK MODES

ELECTRONIC CODE BOOK.
CIPHER BLOCK CHAINING.
COUNTER MODE.

INTRODUCTION: CIPHER BLOCK MODES.

At a basic level a cipher block mode is simply a way of joining together the operations of a block cipher on pieces of data to encrypt an arbitrary length of ciphertext, this is essential as because as the name suggests a block cipher is only able to encrypt a fixed sized block of input and to correctly encrypt a message that is longer than the input size of a block cipher, repeated application the cipher must be used in an iterative way.

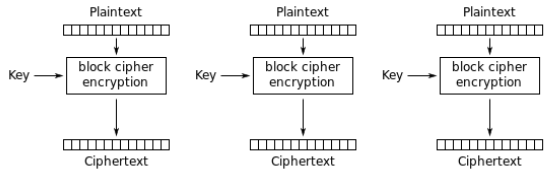
To help commit the concept of a cipher block mode to memory an analogy can be drawn between cipher block modes and the musical modes of the major and minor scales for anybody who is familiar with them, in the same way the major scale can begin on a different note to the root tone creating a mode of the scale and thus changing the perception of its characteristic sound or ‘flavour’, so to can cipher block encryptions be chained together in various combinations to achieve desired properties such as parallelisation.

ELECTRONIC CODE BOOK (ECB).

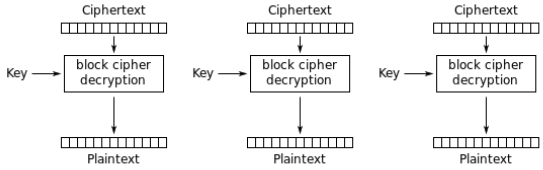
Electronic code book (ECB) is the simplest encryption mode representing perhaps the most obvious approach to repeated application of a block cipher, it entails two steps,

1) Plaintext is split into a number of blocks.
2) Each block is encrypted with the same key using a chosen encryption method.

ECB has a major disadvantage in that identical plaintext blocks are encrypted to identical ciphertext blocks, thus, it has poor entropy properties and does not provide strong message confidentiality, hence it is not recommended for use.



Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

Example: ECB Mode encryption using Vigenere cipher.

Plaintext: “Two all beef patties, special sauce, lettuce, cheese, pickles, onions on a sesame seed bun”.

Encryption: (Vigenere cipher using key: QWERT)

1. **Divide plaintext up into blocks :**
   TWOAL LBEEF PATTI ESSPE CIALS AUCEL ETTUC ECHEE SEPIC KLESO NIONS ONASE SAMES EEDBU N
2. **Pad if necessary :**
   TWOAL LBEEF PATTI ESSPE CIALS AUCEL ETTUC ECHEE SEPIC KLESO NIONS ONASE SAMES EEDBU NXXXX
3. **Encrypt each block separately and append:**
   JSSRE BXIVY FWXKB UOWGX SEECL QQGVE UPXLV UYLVX IATZV AHIJH DESEL EJEJX IWQVL UAHSN DTBOQ

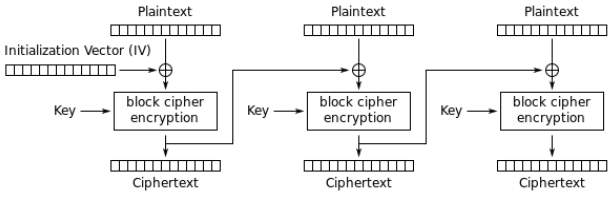
CIPHER BLOCK CHAINING (CBC)

CBC mode adds a feedback mechanism to the ECB mode, the results of previous blocks is fed back into the encryption of the current block, therefore each ciphertext block is dependent not just on the plaintext block that generated it but on all previous plaintext blocks, we can imagine analogy for the CBC mode in cooking, similar to how one can utilise the leftovers of one meal for a subsequent meal and continue this process repeatedly, so to can the outputs of individual encryption blocks be chained together. This chaining process solves the limitations of identical encryption output of identical blocks produced by ECB mode.

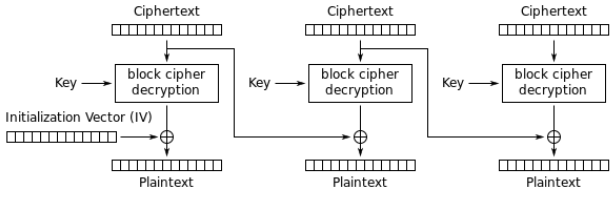
Steps in encryption,

1. Use initialization vector to XOR the first block of plaintext.
2. Encrypt the first block with the key.
3. Use the outputted ciphertext to XOR with the next block.
4. Continue step 3 iteratively on each successive block.

Decryption is straightforward with a cipher text block decrypted normally then stored in a feedback register, after the next block is decrypted it is XORed with the result stored in the feedback register.



Cipher Block Chaining (CBC) mode encryption



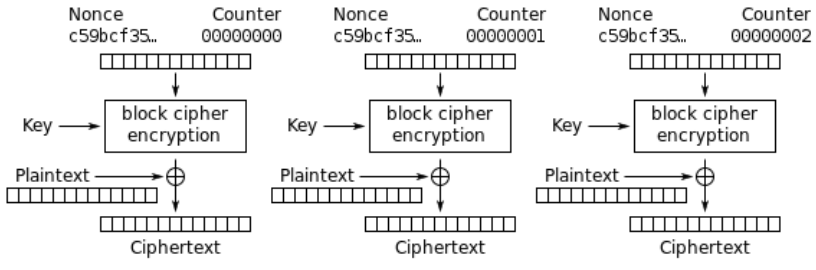
Cipher Block Chaining (CBC) mode decryption

COUNTER MODE (CTR)

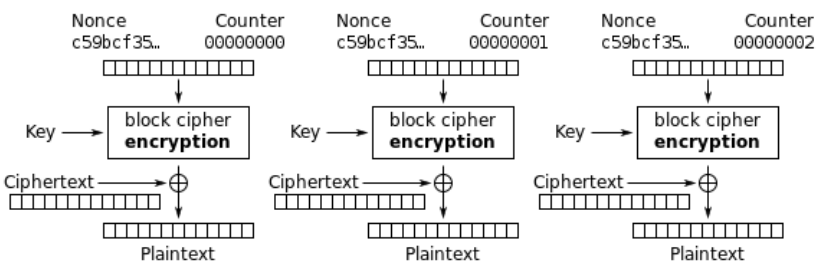
Block ciphers in the counter mode use sequence numbers as the input to the encryption algorithm for each block rather than reuse the output of previous blocks for subsequent encryptions.

Encryption is achieved by splitting the message up into blocks then XORing the plaintext block against a unique nonce generated from the output of the block cipher. The unique nonce for each plaintext block is typically generated in a iterative way by using a counter starting on a randomly generated initial value, the first nonce is generated by encrypting the key against the random initialisation value of the counter, subsequent nonce generation is made by simply incrementing the counter by a fixed constant value (typically 1) and inputing the current iteration of the counter value into the block cipher along with the key.

The encryption method behind Counter Mode is based the randomness of the output of the XOR function and the nonce. Comparatively, ECB Mode only uses a key encryption to produce a ciphertext.



Counter (CTR) mode encryption



Counter (CTR) mode decryption

TECHNICAL SKILLS PRACTICE

WEEK 8

HASHING

PRE-IMAGE ATTACK.
SECOND PRE-IMAGE ATTACK.
COLLISION ATTACK.

INTRODUCTION: HASHING.

A Hash function is any function that takes a variable sized string of data (called the **Pre-image**) maps it into a unique data packet of fixed sized (called the **Hash-value**), In some sense, the resultant output of a hash function provides a succinct summary of the data it hashes.

We can imagine an analogy to how a hash function operates in the real world by comparison to our fingerprints, Human fingerprints are considered to be unique amongst individual people and in some sense can be used to identify and distinguish us uniquely versus other people by comparing the small packet of lines and ridges that make up our fingerprints.

PRE-IMAGE ATTACK.

A **Pre-image attack** on a cryptographic hash function tries to find a message that has a specific hash value.

Pre-image resistance for a cryptographic hash function is the property that for all pre-specified outputs it should be computationally infeasible to find any input that hashes to that output. Ie given a hash of a message y and a hash function h, it should be difficult to find a message m such that y = h(m).

If we relate this back to our fingerprint analogy, pre-image resistance would be comparable to the situation that if we found a fingerprint, it would be difficult to find the person that matched that given fingerprint.

SECOND PRE-IMAGE ATTACK.

Second Pre-Image resistance for a cryptographic hash function h is the property that **given an input m** it should be **difficult to find a second input n** that has the same output as that of a specified input. ie given m it should be difficult to find a second pre-image $n \neq m$ such that $h(m) = h(n)$.

We could liken a second pre-image attack back to our fingerprint analogy in that it is comparable to **picking a specific person in a group** then trying to find **another person** in the group such that the two people have the same fingerprints, however this is a little bit of an abuse of the fingerprint analogy as humans are understood to have unique fingerprints.

COLLISION ATTACKS.

Collision resistance for a cryptographic hash function h is the property that it should be difficult to find two different messages m and n , such that h(m) = h(n).

Collision resistance implies second pre-image resistance but does not guarantee pre-image resistance.

Every cryptographic hash function is inherently vulnerable to collisions using a **birthday attack**, which is simply the method of finding a collision by evaluating the hash function for different input values chosen randomly or pseudorandomly until the same result is found more than once.

We could liken a collision attack back to our fingerprint analogy in that it is comparable to finding **any two** people in a group such that they have the same fingerprints, although again this is a little bit of an abuse of the fingerprint analogy.

Because of the mathematics of the birthday problem, finding hash collisions using a birthday attack can be found rather efficiently, with a hash of n bits able to be broken in $2^{n/2}$ evaluations of the hash function using a birthday attack.

EXAMPLE: COLLISIONS USING BIRTHDAY ATTACK

On average how many hash computations would be required to perform a successful collision on a 16-bit hash using a birthday attack?
[Give your answer to the nearest power of two to the number you believe is the correct value. So if you thought the correct value was 42 you would enter 32 as your answer since that is closer to 42 than 64 is.]

The answer is 256 computations as the square root of 2^16 = 2^8=256.

TECHNICAL SKILLS PRACTICE APPENDIX

DIFFIE-HELLMAN WORKED EXAMPLE.
TECHNICAL RESTRICTIONS ON RSA.
RSA WORKED EXAMPLE.

DIFFIE HELLMAN WORKED EXAMPLE

Calculate the shared secret key generated in a Diffie-Hellman key exchange between Alice and Bob using using modulus p=101, and generator g=12, such that Alice choses secret number a=8, and Bob chooses secret number b=19.

Solution.

1) Alice and bob share the modulus p=101 and the generator g=12 publicly.
2) Alice sends x=g^a=12^8 =52(mod 101) to Bob.
3) Bob sends the message y=g^b=12^19=50(mod 101) to Alice.
4) They both calculate the common key k, where k=x^b=50^8=y^a=52^19 = 58 (mod 101).

The shared secret key generated in this example is k=58 (mod 101).

TECHINICAL RESTRICTIONS ON RSA.

In practice RSA has a number of technical restrictions on the choice of input into the algorithm, these are not essential to memorise for the exam but may be of interest to students.

1) p,q should not be too close in size (otherwise Fermat factoring can be applied).
2) gcd(p-1,q-1) should be small, where gcd is the greatest common divisor of two numbers.
3) $p \pm 1$, $q \pm 1$ should have large prime factors or n can be factored by either pollard's p-1 or William's p+1 algorithm.
4) If d or e is small, try another e.
5) Neither of the primes p or q should be used repeatedly or they can be found by calculating the gcd or public keys.

RSA WORKED EXAMPLE

Show an RSA key generation calculation using the primes p=3 and q=11 and public encryption function e=7, then show the encryption and decryption of the character B using the following standard encoding (to avoid A being a small number).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

Solution.

1) Calculate n=p*q =3*11= 33.
2) Calculate $\phi(n)= \phi(p)*\phi(q)= 2*10=20$.
3) Find d the modular multiplicative inverse of e, by solving the equation d*e=d*7=1(mod 20) , by brute force we find 3*7=21=1(mod 20) , therefore d=3.

4) public key is (33,7), private key is(33,3).

If someone wanted to encrypt the character B, they would encode then encipher via, B-> 4 , $E_7(4)$ = 4^7 =16 (mod 33) -> N.

Someone receiving the message N would then decipher by doing the following N-> 16, $D_3(16)$ = 16^3 (mod 33) = 4(mod 33) -> B.