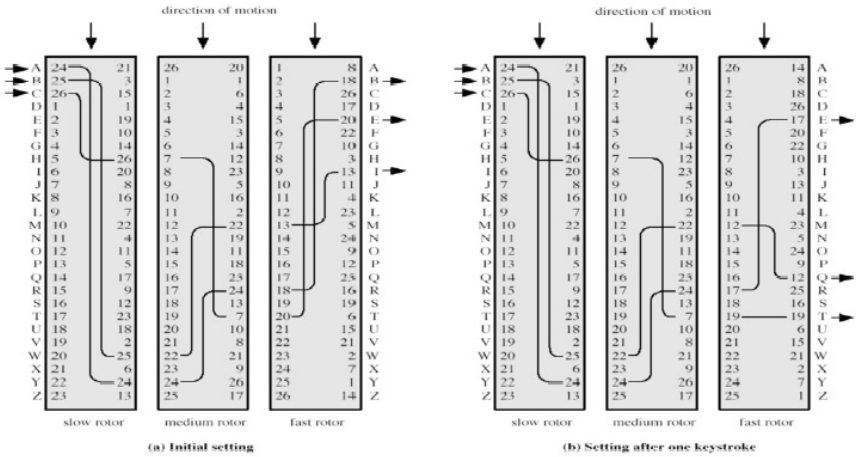
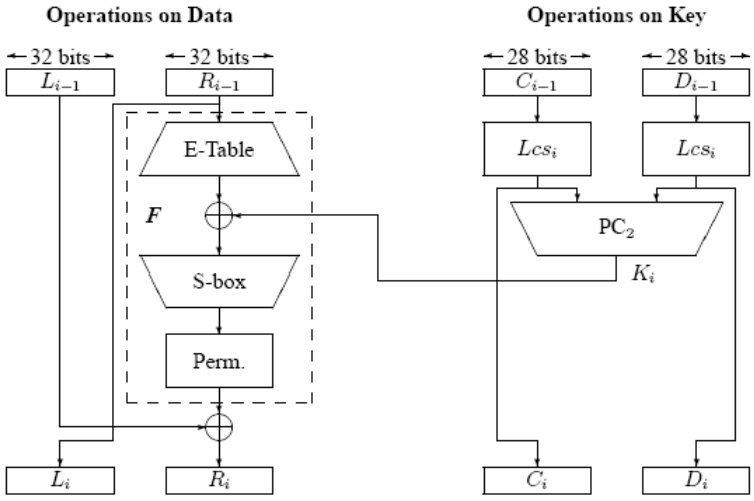
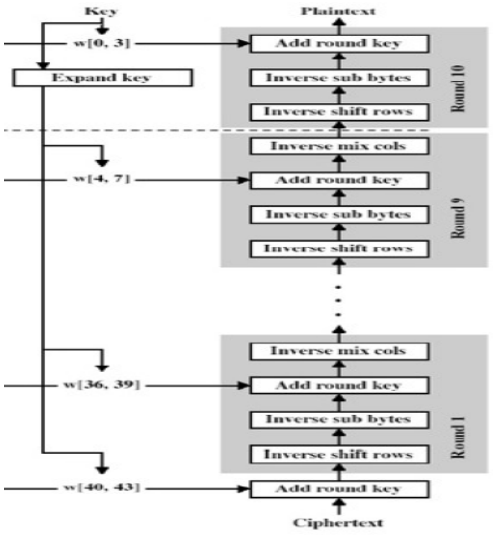
EXAMINATION MARKING SCHEME
Module: Internetwork Security (EE548)
Date: 14-03-2008

<table>
<tr><th>Q#</th><th></th><th>MARK</th></tr>
<tr><td>Q1a</td><td>(i) A Security service is a service that enhances the security of a data processing systems and information transfers.
(ii) A Security mechanism is a mechanism that is designed to detect, prevent or recover from a security attack.
(iii) Non-repudiation requires that neither the sender nor receiver of a message be able to deny the transmission.
(iv) Authentication requires that the origin of a message be correctly identified with the assurance that the identity is not false.</td><td>[4 Marks]</td></tr>
<tr><td>Q1b</td><td>(i)
<table><tr><td>Mu - uz
St – tb
Se - dl
Ey - gz
Ou - pn</td><td>Ov - nw
Er - lg
Ca - tg
Do - tu
Ga - er</td><td>Nw - ov
es - ld
tc - bd
om - uh
in - fp</td><td>ga - er
to - hw
nc - qs
ex - rz</td></tr></table></td><td>[6 Marks]</td></tr>
<tr><td></td><td>(ii)
<table><tr><td>L</td><td>A</td><td>R</td><td>G</td><td>E</td></tr><tr><td>S</td><td>T</td><td>B</td><td>C</td><td>D</td></tr><tr><td>F</td><td>H</td><td>I/J</td><td>K</td><td>M</td></tr><tr><td>N</td><td>O</td><td>P</td><td>Q</td><td>U</td></tr><tr><td>V</td><td>W</td><td>X</td><td>Y</td><td>Z</td></tr></table></td><td>[4 Marks]</td></tr>
<tr><td></td><td>(iii)

The two matrices produce the same ciphertext. Any matrix consisting of simple row or column shifts of the original will produce the same ciphertext. This is due to the fact that the matrix is viewed as a circular array on the top and bottom as well as on the sides.</td><td>[5 Marks]</td></tr>
</table>

Q1c	 The system consists of a number of independently rotating cylinders through which electrical pulses can flow. Each cylinder has 26 input pins and 26 output pins with internal wiring that connects each input to a unique output, Now, consider a machine consisting of one cylinder. After each input is depressed, the cylinder moves one position thus creating a different monoalphabetic cipher. After 26 letters of plaintext (a full rotation) the cylinder is back to its original position so we have a polyalphabetic substitution cipher of period 26. A single cylinder is trivial but the use of multiple cylinder is formidable.	[6 Marks]
Q2a	(i)  Figure 2.3: Details of a single DES round.	[7 Marks]

(ii)



[7 Marks]

(iii)

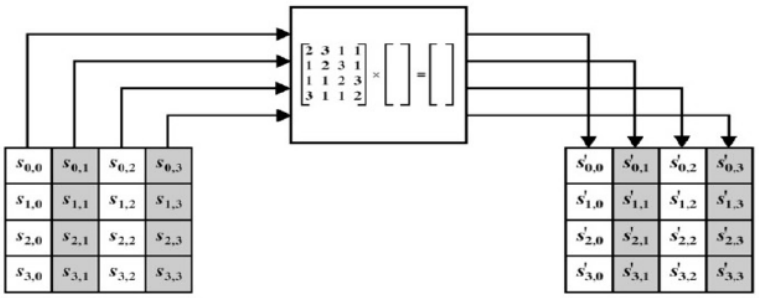
The MixColumn stage is basically a substitution but it makes use of arithmetic of GF(2^8). Each column is operated on individually. Each byte of a column is mapped into a new value that is a function of all four bytes in the column. The transformation can be determined by the following matrix multiplication on state

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Each element of the product matrix is the sum of products of elements of one row and one column. In this case the individual additions and multiplications are performed in GF(2^8). The MixColumns transformation of a single column j (0 _ j _ 3) of state can be expressed as:

$$\begin{aligned} s'_{0,j} &= (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j}) \\ s'_{3,j} &= (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j}) \end{aligned}$$

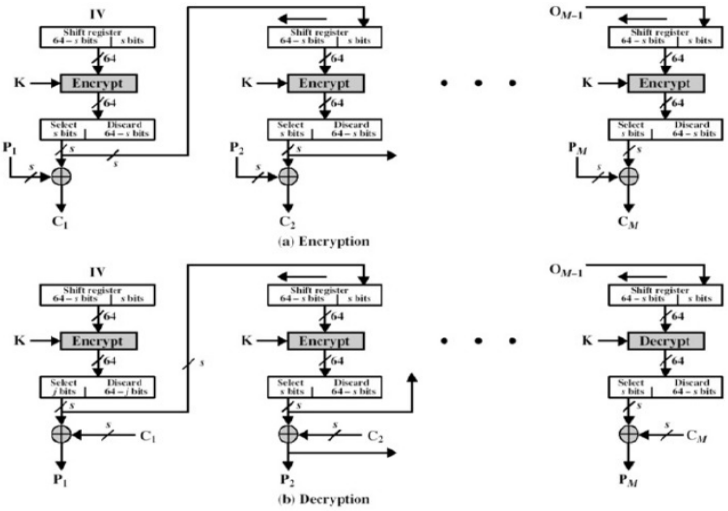
where • denotes multiplication over the finite field GF(2^8).



[5 Marks]

Q2b

Output Feedback (OFB) Mode



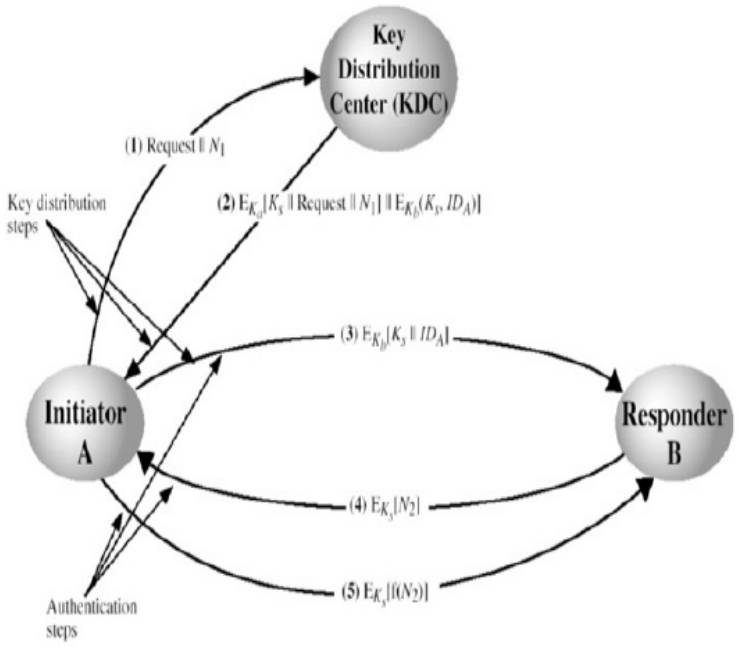
[5 Marks]

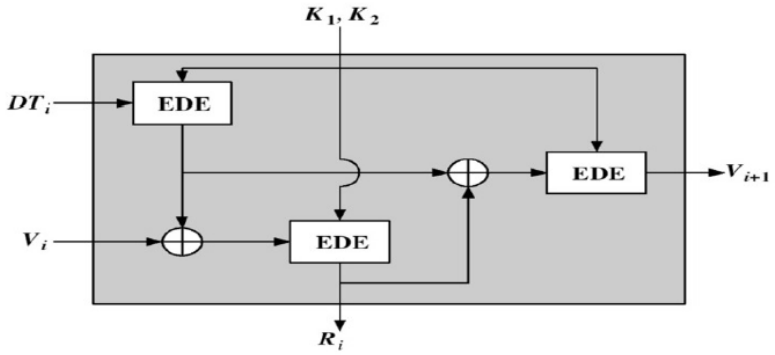
Figure 7.13: Output Feedback (OFB)

The fact that CFB, OFB and Counter modes use only encryption and not decryption influenced the design of the AES.

[1 Marks]

Q3a



Q3b	1. A requests a session key from the KDC. The message includes the identity of A and B and a unique identifier N1 for this transaction - called a "nonce". This could be a timestamp or random number and it is desirable that it be difficult to guess. 2. The KDC responds with a message encrypted using Ka thus only A can decode it. It contains two items intended for A, the one-time session key Ks and the original request message including the nonce. The latter allows a match to be made between this response and the request. With this data, A can verify that its original request was not altered. As well as these two items, it includes another two items intended for B but encrypted using Kb: the session key Ks, and an identifier of A (its network address) IDA. 3. A stores the session key Ks and forwards the information encrypted with Kb to B: EKb [Ks||IDA] where || denotes concatenation. B now knows the session key Ks, that the other party is A, and that the information originated at the KDC (because it was encrypted using Kb). At this point Ks has been delivered securely to A and B for their session. Two additional steps in the process are desirable: 4. Using the newly minted session key for encryption, B sends a nonce N2 to A. 5. Also, using Ks, A responds with f(N2) where f is a function transforming N2 (say add one).	[12 Marks]
	 The algorithm makes use of triple DES and has the following: – Input: Two pseudorandom inputs drive the algorithm. One is a 64 bit representation of the current date and time updated on each no. generation. The other is a 64 bit seed value initialized to some arbitrary value and updated during the generation process. – Keys: The generator makes use of three triple DES encryption modules. All three make use of the same pair of 56 bit keys, which must be kept secret. – Output: These are a 64 bit pseudorandom no. (Ri) and a 64 bit seed value (Vi+1).	[7 Marks]

Q3c

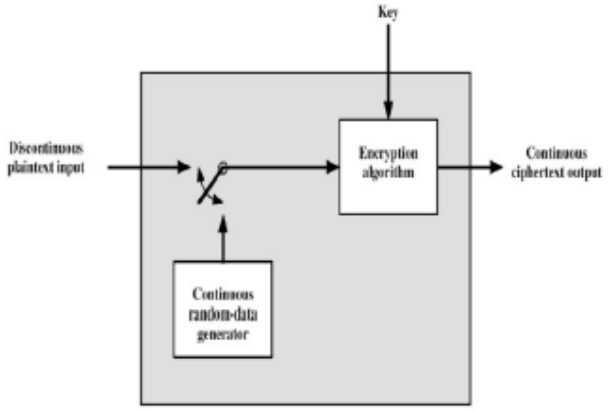
(i)

1. The function should be a full-period generating function (i.e. should generate all numbers between 0 and m before repeating).
2. The generated sequence should appear random. The sequence is not random at all but there is a wide variety of tests to assess the degree of randomness exhibited.
3. The function should implement efficiently with 32-bit arithmetic.

[3 Marks]

(ii)

The output produces ciphertext continuously when no input is available as the random number generator takes over producing a series of encrypted random numbers. Shown in figure.



[3 Marks]

Q4a

(i)

A **Group** $\{G, \cdot\}$ is a set under some operation $(\cdot)$[3] if it satisfies the following 4 axioms:

1. **Closure** (A_1): For any two elements $a, b \in G,\ c = a \cdot b \in G$
2. **Associativity** (A_2): For any three elements $a, b, c \in G,\ (a \cdot b) \cdot c = a \cdot (b \cdot c)$
3. **Identity** (A_3): There exists an **Identity** element $e \in G$ such that $\forall_{a \in G},\ a \cdot e = e \cdot a = a$.
4. **Inverse** (A_4): Each element in G has an inverse i.e[4]., $\forall_{a \in G}\ \exists_{a^{-1} \in G},\ a \cdot a^{-1} = a^{-1} \cdot a = e$.

If in addition the set follows the axiom:

5. **Commutativity** (A_5): For any $a, b \in G, a \cdot b = b \cdot a$.

it is then said to be an **Abelian group**.

(ii)

A **Ring** $\{R_g, +, \times\}$ is a set with two binary operations[5] *addition* and *multiplication* that satisfies the following axioms:

1. **Abelian Group under addition** $(A_1 \rightarrow A_5)$: It satisfies all of the axioms for an Abelian group (all of the above) with the operation of *addition*. The identity element is 0 and the inverse is denoted $-a$.
2. **Closure under multiplication** (M_1): For any two elements $a, b \in R_g$, $c = ab \in R_g$
3. **Associativity of multiplication** (M_2): For any elements $a, b, c \in R_g$, $(ab)c = a(bc)$
4. **Distributive** (M_3): For any elements $a, b, c \in R_g$, $a(b + c) = ab + ac$

If in addition the Ring follows the axiom:

5. **Commutativity** (M_4): For any $a, b \in R_g$, $ab = ba$.

it is then said to be a **commutative ring**.

(iii) A number is said to be co-prime if it's only positive divisors are itself and 1. Another name is relatively prime.

(iv) A least residue is the remainder left when one number is operated on modulo some other number.

(v) A residue class is the set of numbers congruent to a least residue modulo m. [12 Marks]

Q4b

The result can be shown to be {37} and m(x) is not needed in this case. [4 Marks]

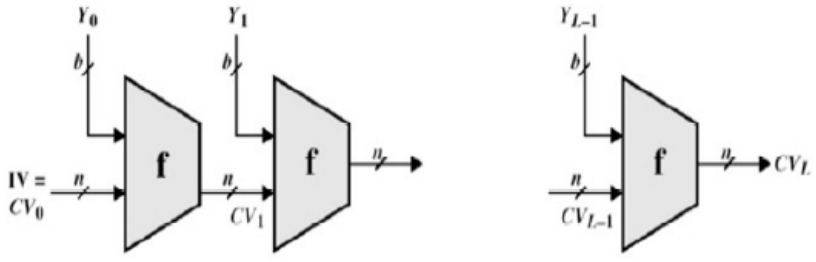
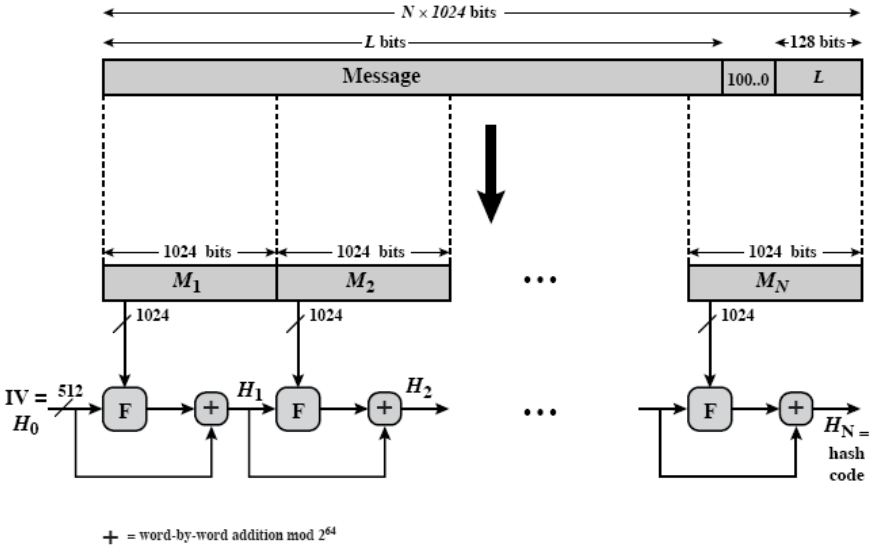
The value of m(x) given here is that used in AES. [1 Marks]

Q4c

- The set of all value less than n=pq is {1, pq-1}
- The set of multiples of p is {p, 2p,…,(q-1)p}
- The set of multiples of q is {q, 2q,…,(p-1)q}
- The set relatively prime to n is {1,pq-1} – {p, 2p,…,(q-1)p} - {q, 2q,…,(p-1)q}
- The number of the above set is (pq-1) – (q-1) – (p-1) = (p-1)(q-1)

[8 Marks]

Q#		MARK
Q5a	(i) 1. H can be applied to a block of data of any size. 2. H produces a fixed-length output. 3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical. 4. For any given code h, it is computationally infeasible to find x such that H(x) =h. 5. For any given block x, it is computationally infeasible to find y 6= x with H(y) = H(x) (sometimes referred to as **weak collision property**). 6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y) (sometimes referred to as **strong collision property**). (ii) 1. The source, A is prepared to "sign" a message by appending the appropriate m-bit hash code and encrypting that hash code with A's private key. 2. The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning. The opponent prepares an equal number of messages, all of which are variations of the fraudulent message to be substituted for the real one. 3. The two sets of messages are compared to find a pair of messages that produce the same hash code. The probability of success is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made. 4. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. 5. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known. (i)	[6 Marks] [5 Marks]

Q5b	 IV = Initial value CV = chaining variable Y_i = ith input block f = compression algorithm L = number of input blocks n = length of hash code b = length of input block (ii) 	[7 Marks] [7 Marks]

Date: 14-03-2008

Q#		MARK
Q6a	In order to carry out a DPA attack, a method must be devised to produce a random set of J plaintext inputs that can be sent to the cryptosystem for encryption[1]. On receiving these plaintext inputs, pi_j, $1 \leq j \leq J$, the board will begin to run its algorithm and draw varying amounts of power. These power fluctuations can be sampled using a digital sampling oscilloscope which should be capable of sampling at about 20-30 times the clock frequency being used. The waveforms observed for each pi_j can be represented as a matrix wf_{jk}[2], where $1 \leq k \leq K$. A second column matrix, co_j, can also be used to represent the ciphertext output. In practice, each row of wf_{jk} would probably be stored as a separate file for ease of processing. Having captured each power waveform and ciphertext output, a function known as a *partitioning function*, D(.), must now be defined. This function will allow division of the matrix wf_{jk} into two sub-matrices $wf0_{pk}$ and $wf1_{qk}$ containing P and Q rows respectively, with $1 \leq p \leq P$ and $1 \leq q \leq Q$ where $P + Q = J$. Provided that the inputs pi_j were randomly produced, then $P = Q = J/2$ as $J \rightarrow \infty$ (i.e. the waveforms will be divided equally between the two sets). The partitioning function allows the division of wf_{jk} because it calculates the value of a particular bit, at particular times, during the operation of the algorithm. If the value of this bit is known, then it will also be known whether or not a power bias should have occurred in the captured waveform. For a 1, a bias should occur, and for a 0 it shouldn't. Separating the waveforms into two separate matrices (one in which the bias occurred and another in which it didn't) will allow averaging to reduce the noise and enhance the bias (if it occurred). For randomly chosen plaintexts, the output of the D(.) function will equal either a 1 or 0 with probability $\frac{1}{2}$ (this is just another statement of the fact that $P = Q = J/2$ as $J \rightarrow \infty$). An example of a partitioning function is: $$D(C_1, C_6, K_{16}) = C_1 \oplus SBOX1(C_6 \oplus K_{16}) \quad (11)$$ where SBOX1(.) is a function that outputs the target bit of S-box 1 in the last round of DES (in this case it's the first bit), C_1 is the one bit of co_j that is exclusive OR'ed with this bit, C_6 is the 6 bits of co_j that is exclusive OR'ed with the last rounds subkey and K_{16} is the 6 bits of the last round's subkey that is input into S-box 1. The value of this partitioning function must be calculated at some point throughout the algorithm. So, if the values C_1, C_6 and K_{16} can be determined, it will be known whether or not a power bias occurred in each waveform. It is assumed that the values C_1 and C_6 can be determined and the value of the subkey K_{16} is the information sought. To find this, an exhaustive search needs to be carried out. As it is 6 bits long, a total of $2^6 = 64$ subkeys will need to be tested. The right one will produce the correct value of the partitioning bit for every plaintext input. However, the incorrect one will only produce the correct result with probability $\frac{1}{2}$. In this case, the two sets $wf0_{pk}$ and $wf1_{qk}$ will contain a randomly[3] distributed collection of waveforms which will average [1] This method must be automated as the number of random plaintext inputs will be quite large. [2] The subscripts j and k are used to identify the plaintext number causing the waveform and the time sample point within that particular waveform, respectively. [3] Provided the plaintext inputs are randomly chosen.	

Date: 14-3-2008

Q#		MARK

out to the same result. The differential trace (discussed below) will thus show a power bias for the correct key only. Of course it means that 64 differential traces are needed but this is a vast improvement over a brute force search of the entire 56 bit key.
Mathematically, the partitioning of wf_{jk} can be represented as

$$wf0_{pk} = \{wf_{jk} | D(.) = 0\} \tag{12}$$

and

$$wf1_{qk} = \{wf_{jk} | D(.) = 1\} \tag{13}$$

Once the matrices $wf0_{pk}$ and $wf1_{qk}$ have been set up, the average of each is then taken producing two waveforms $awf0_k$ and $awf1_k$ both consisting of K samples. By taking the averages of each, the noise gets reduced to very small levels but the power spikes in $wf1_{pk}$ will be reinforced. However, averaging will not reduce any periodic noise contained within the power waveforms and inherent to the operations on the cryptographic board. This can largely be eliminated by subtracting $awf0_{pk}$ from $awf1_{qk}$ (this can be thought of as demodulating a modulated signal to reveal the "baseband", where the periodic noise is the "carrier"). The only waveform remaining will be the one with a number of bias points identifying the positions where the target bit was manipulated. This trace is known as a *differential trace*, ΔD_k.
Again, in mathematical terms, the above can be stated as

$$awf0_k = \frac{1}{P} \sum_{wf_{jk} \in wf1} wf_{jk} = \frac{1}{P} \sum_{p=1}^{P} wf0_{pk} \tag{14}$$

and

$$awf1_k = \frac{1}{Q} \sum_{wf_{jk} \in wf0} wf_{jk} = \frac{1}{Q} \sum_{q=1}^{Q} wf1_{qk} \tag{15}$$

The differential trace ΔD_k is then obtained as

$$\Delta D_k = awf1_k - awf0_k \tag{16}$$

The last five equations can now be condensed into one:

$$\Delta D_k = \frac{\sum_{k=1}^{K} D(.) wf_{jk}}{\sum_{k=1}^{K} D(.)} - \frac{\sum_{k=1}^{K} (1 - D(.)) wf_{jk}}{\sum_{k=1}^{K} (1 - D(.))} \tag{17}$$

As $J \to \infty$, the power biases will average out to a value ϵ which will occur at times k_D - each time the target bit D was manipulated. In this limit, the averages $awf0_k$ and $awf1_k$ will tend toward the expectation $E\{wf0_k\}$ and $E\{wf1_k\}$, and equations 16 and 17 will converge to

$$E\{wf1_k\} - E\{wf0_k\} = \epsilon, \qquad \text{at times} \quad k = k_D \tag{18}$$

and

$$E\{wf1_k\} - E\{wf0_k\} = 0, \qquad \text{at times} \quad k \neq k_D \tag{19}$$

Therefore, at times $k = k_D$, there will be a power bias ϵ visible in the differential trace. At all other times, the power will be independent of the target bit and the differential trace will tend towards 0.
The above will only work if the subkey guess was correct. For all other guesses the partitioning function will separate the waveforms randomly, and equations 18 and 19 will condense to

$$E\{wf1_k\} - E\{wf0_k\} = 0, \qquad \forall k \tag{20}$$

As mentioned above, 64 differential traces are needed to determine which key is the correct one. Theoretically, the one containing bias spikes will allow determination of the correct key however, in reality the other waveforms will contain small spikes due to factors such as non-random choices of plaintext inputs, statistical biases in the S-boxes and a non-infinite number of waveforms collected. Generally however, the correct key will show the largest bias spikes and can still be determined quite easily.
The other 42 bits from the last round's subkey can be determined by applying the same method to the other 7 S-boxes[4]. A brute force search can then be used to obtain the remaining 8 bits of the 56 bit key.

[13 Marks]

Q6b	The following could be used as mitigation techniques for power attacks in general: 1. Timing Randomisation: This involves placing random time delays into the software so that a power analysis will not be possible. With random delays introduced, a steady trigger will not be sufficient to allow the averaging to work and will therefore act as a countermeasure. 2. Internal power supplies/power supply filtering: This would be another method that could be used to reduce the possibility of a power attack. For example, Adi Shamir proposes building a simple capacitance network into each smartcard to allow the fluctuations to be contained within the smartcard itself thereby preventing power attacks. 3. Data masking: One of the methods proposed consists of masking the intermediate data (i.e. mask the input data and key before executing the algorithm). This would make the power fluctuations independent of the actual data. 4. Tamper Resistance: This involves placing some detection/prevention system around the device to stop intruders gaining access to the power fluctuations. 5. Fail Counters: A differential power analysis attack requires the attacker to obtain a significant number of power waveforms. In order to do this the attacker must have the ability to run quite a few encryptions on the system under attack. If the number of encryptions were limited to a certain number then the attacks would become increasingly difficult. 6. Removal of conditional elements: One of the main features used to attack the square and multiply algorithm is the fact that it has a conditional multiplication that depends on the value of the exponent bit being operated upon. One suggested countermeasure is to implement this multiplication in every round (regardless of the value of the bit) and to only do a register update when the bit is a 1.	[8 Marks]
Q6c	1. Unconditionally Secure defines a system where the ciphertext generated does not contain enough information to determine uniquely the corresponding plaintext no matter how much ciphertext is available or how much computation power the attacker has. 2. Computationally Secure defines a system where the cost of breaking the cipher exceeds the value of the encrypted information and the time required to break the cipher exceeds the useful lifetime of the information 3. A cryptanalytic attack is one that tries to attack the mathematical weaknesses of the algorithm 4. An implementation attack is one that tries to attack the actual implementation of the cryptographic system such as a smart card.	[4 Marks]