Review Questions CH6 Assignment 02 | 214101058

6.7 Shift Rows

The forward shift row transformation is called Shift Rows in AES.

The first row of state is not altered.

For the second row, a 1-byte circular left shift is performed.

For the ~~second~~ third row, a 2byte circular left shift is performed

For the fourth row, a 3 byte circular left shift is performed.

In the Shift Rows phase of AES, each row of the 128-bit internal state of the cipher is shifted.

The row in this stage refer to the standard representation of the internal state in AES, which is a 4x4 matrix where each cell contain a byte.

6.9 Mix Columns

The forward mix columns transformation called Mix Columns, operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column.

Each element in the product matrix is the sum of products of elements of one row and one column.

The individual additions and multiplications are performed in $GF(2^8)$

This multiplication has the property of operating independently over each of the columns of initial matrix.

RQ 6.11 Key Expansion Algorithm

4W × 4B

The AES key Expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 B).

This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher

The key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time.

Each added word w[i] depends on the immediately preceding word, w[i-1] and the word four positions back w[i-4].

In three out of 4 cases, a simple XOR is used. For a word whose position in the w array is multiple of 4, a more complex function is used.

The complex function consists of following subfunction.

1. RotWord performs a one-byte circular left shift on a word.
2. SubWord performs a byte substitution on each byte of its input word, using the S-box.
3. The result of step 1 and 2 is XORed with a round constant, Rcon[j].

Chp 7 Review Questions

RQ 7.4 Block cipher modes of operation

Encryption algorithms are divided into two categories based on the input type as a block cipher and stream cipher.

Block Cipher is an encryption algorithm that takes a fixed size of input say b bits and produces a cipher text of b bits again. If the input is larger than b bits it can be divided further.

For different applications and uses, there are several modes of operations for a block cipher.

a) Electronic Code book:

It is a simplest mode in which plain text is handled one block at a time and each block of plaintext is encrypted using the same key.

The term codebook is used because, for a given key, there is a unique ciphertext for every b-bit block of plaintext. For a message larger than b-bits, the procedure is simply to break the message into b-bit blocks, padding the last block if necessary. Decryption is performed one block at a time, always using the same key.

b) Cipher Block Chaining Mode:

CBC is an advancement mode on ECB. It overcome the security deficiency of ECB. In CBC, if the same plaintext block, if repeated, produces different cipher blocks.

In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding cipher block, the same key is used for each block.

(c) Ciphet Feedback Mode

Input is ~~processed~~ processed s bits at a time. In this mode, the cipher is given as feedback to the next block of encryption with some new specifications: first an initial vector IV is used for first encryption and output bits are divided as a set of 's' and 'b-s' bits.

The left-hand side s bits are selected along with plaintext bits to which an XOR operation is applied. The result is given as input to a shift register having b-s bits to LHS, s bits to RHS and the process continues.

(d) Output Feedback Mode.

It is ~~same~~ similar in structure to that of CFB. For OFB, the output of the encryption function is fedback to became the input for encrypting the next block of plain text, instead of the actual cipher of XOR output.

In OFB mode operates on full blocks of plaintext and ciphertext instead of operating on s-bit subset of CFB mode.

In this bit errors in transmission do not propagate. and decreases the dependency or relationship of the cipher on the plaintext

(e) Counter Mode

CTR is a counter-based block cipher implementation. A counter equal to the plaintext block size is used. The counter is initialized to some value. and then incremented by 1 for each subsequent block. (modulo 2^b, b is block size)

For encryption, the counter is encrypted and then XORed

with the plaintext block to produce the cipher text block; there is no chaining.

For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block.

CHAPTER 8

RQ 8.3 / Stream Cipher and one-time pad.

The stream cipher is similar to the one-time pad. The difference is that one-time pad uses a genuine random number stream, whereas a stream cipher uses a pseudorandom number stream.

→ ~~It is used~~ One-time pad is used to encrypt small, human-generated message, while modern stream cipher are used to encrypt computer-generated data streams

→ One-time pad is a stream ciphers, stream ciphers are not one-time pads

→ The most important difference is in their security level. One time Pad has a perfect secrecy but stream ciphers have computational secrecy.

CHAPTER 9

RQ 9.2 Roles of Public & Private keys.

A public key and a private key are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

All participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed.

Public keys: It is used to encrypt and a private key is used to decrypt the data. The private key is shared between the sender and receiver of the encrypted sensitive information.

The Public key is also called asymmetric cryptography.

Private key: It is used to both encrypt and decrypt the data. The key is shared between the sender & receiver of the encrypted sensitive information. The private key is also called symmetric being common for both parties.

Private key cryptography is faster than public key cryptography mechanism

RQ 9.4 Requirements for Public-Key Cryptography.

The conditions the algorithms for Public-key Cryptography must fulfill:

1. It is computationally easy for a Party B to generate a key pair (Public key PU_b, Private key PR_b)
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the

corresponding ciphertext.

$$C = E(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message.

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. It is computationally infeasible for an adversary, knowing the public key, PU_b to determine the private key, PR_b.

5. It is computationally infeasible for an adversary, knowing the public key, PU_b and a ciphertext C, to recover the original message, M.

6. The two keys can be applied in either order.

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

Chapter 10

RQ 10.1 Explain Diffie-Hellman Key Exchange?

A primitive root of a prime number p is one whose powers modulo p generate all the integers from 1 to p-1.

$a \bmod p, a^2 \bmod p, \ldots, a^{p-1} \bmod p$. are distinct and consists of the integers from 1 through p-1 in some permutation.

For any integer b and a primitive root 'a' of prime number p, we can find a unique exponent i such that

$$b \equiv a^i \pmod p \quad \text{where } 0 \le i \le (p-1)$$

The exponent i is refered to as the discrete logarithm of b for the base a. $dlog_{a,p}(b) = i$

Diffie-Hellman Key exchange is a simple public key algorithm.

→ There are two publicly known numbers : a prime q and an integer α that is primitive root of q.

User A

↳ generate a private key X_A such that $X_A < q$

↳ computes $Y_A = \alpha^{X_A} \bmod q$

user B

generate a private key X_B such that $X_B < q$

computes $Y_B = \alpha^{X_B} \bmod q$

each side keeps the X value private and makes the Y value available publicly to other side.

↳ computes key as $K = (Y_B)^{X_A} \bmod q$

computes key as $K = (Y_A)^{X_B} \bmod q$

↳ These two calculations produce identical results

Problems / chapter 6

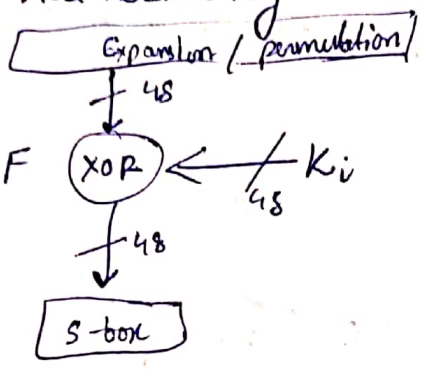
P6.6 c

AES and DES are both examples of symmetric block ciphers but have certain dissimilarities

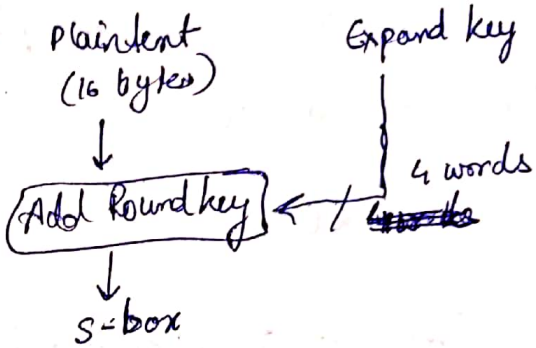
	AES	DES
1.	AES (Advanced Encryption Standard) is Byte Oriented	DES (Data Encryption Standard) is Bit Oriented
2.	Key length can be 128 b, 192 b, & 256 bit.	The key length is 56 bits in DES
3.	# of Rounds depends on key length 10 (128 b), 12 (192 b), 14 (256 b)	DES involves 16 rounds of identical operation
4.	Structure based on substitution–permutation network	Structure is based on Feistel Network.
5.	More Secure	It can be broken easily
6.	Rounds in AES are Byte Substitution, Shift Row, Mix Column, & Key addition	Rounds in DES are : Expansion, XOR operation with round key, substitution & permutation
7.	No known crypt-analytical attacks against AES implementations possible.	known attacks against DES include brute-force, Linear-crypt-analysis & differential crypt-analysis
8.	AES is faster	comparatively slower

6.6(a) DES : XOR of subkey material with the input to the 'f' function

AES → Add Round key



DES



AES

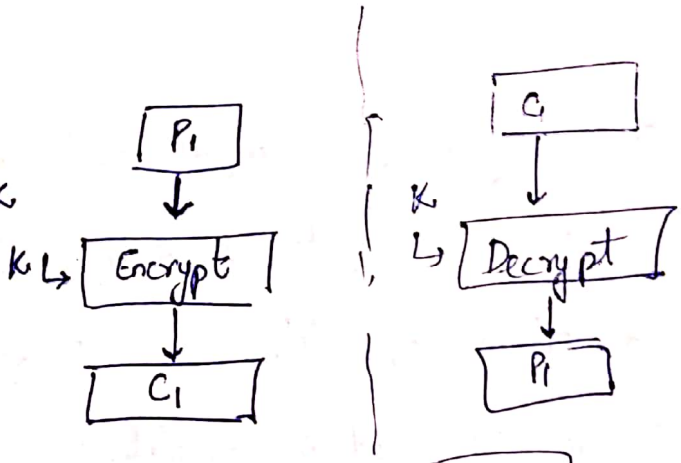
6.6(c) DES : f function

AES : The ByteSub step, because it contributes non-linearity to AES. (Substitution bytes)

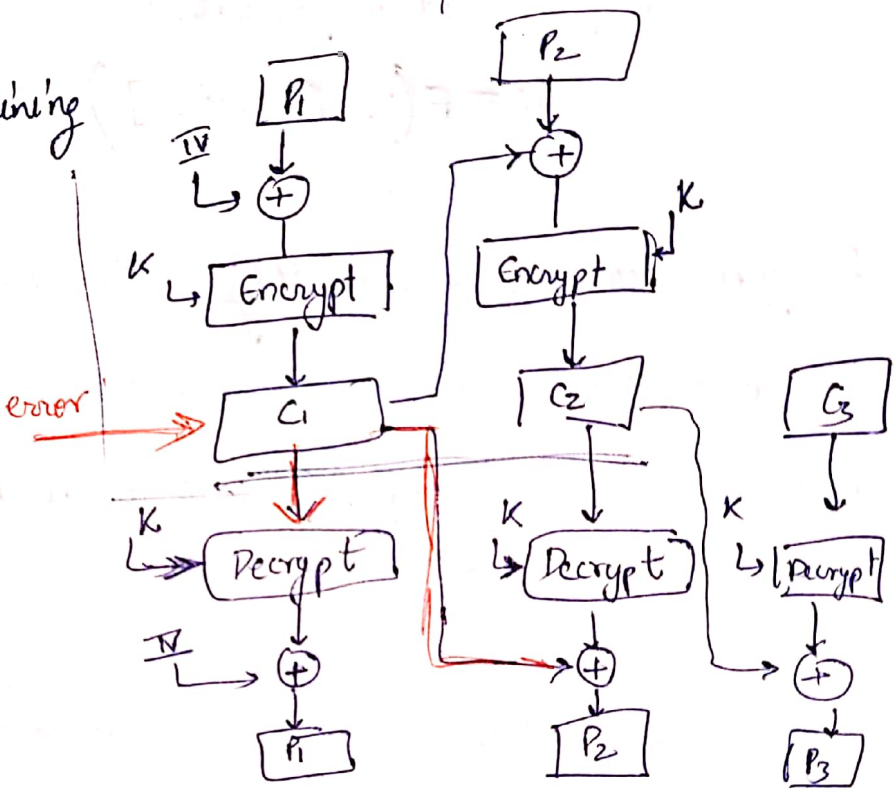
Chapter 7 | Problems

P7.4

ECB – Electronic Codebook



CBC – Cipher Block Chaining



7.4(a)

No. For example, suppose C_1 is corrupted.

The output block P_3 depends only on the input blocks C_2 & C_3

7.5(b)

An error in P_1 definetly affects C_1 but C_1 is also input to the calculation of C_2, C_2 is affected.

This effect will be propagated indefinitely, ~~so that~~ such that all cipher blocks are affected.

However at the receiving end, the decryption algorithm restores the correct plaintext for blocks except the one in error.

Therefore, error only affects the corresponding decrypted Plaintext block.

P7.5

No, In CBC encryption, the input block to produce each cipher block operations (except the first) depends on the result of previous forward cipher operation, so the forward cipher operations cannot be performed in parallel.

$$C_j = E\left(K, [C_{j-1} \oplus P_j]\right)$$

Decryption can be done parallely. There are two basic strategies: one is to treat the message in an interleaved fashion, and the other is to break it up into a single chunk for each parallel process.

P7.8

CFB mode, bit-error in 8-bit CFB transmission of cipher text.

~~[illegible]~~, $s = 8$ bit, $b = 64$ bit (DES)

for 64 bit shift register & s = 8-bit CFS, the error propagates for $64/s = 64/8 = 8$ blocks after the erroneous block 'i'.

Total 9 blocks will be affected.

A bit error in ith transmitted cipher byte will cause bit error at the same bit position in ith recovered plain text byte at the receiver. The erroneous cipher text bytes will then be fed to shift register and cause bit errors in the plain text as long as the erroneous bit stays in the shift register.

Chapter 8

P8.6

RC4 is a stream-cipher and a variable length key algorithm.

We have to use a key of length 255 Bytes in order for RC4 state 'S' to be unchanged during initialization.

The first two bytes $K[0] = 0$

$K[1] = 0$

Subsequently we have $K[2] = 255$, $K[3] = 254$, $K[255] = 2$

Now.

$S[0] = 0$	$K[0] = 0$	$T[0] = 0$
$S[1] = 1$	$K[1] = 0$	$T[1] = K[1] = 0$
$S[2] = 2$	$K[2] = 255$	$T[2] = K[2] = 255$
⋮	⋮	⋮
$S[255] = 255$	$K[255] = 2$	$T[255] = K[255] = 2$

Initial Permutation of S

```
j = 0
for (i = 0 to 255) do
    j = (j + S[i] + T[i]) mod 256 ;
    swap (S[i], S[j]);
```

i = 0 swap(S[0], S[0])

i = 1 swap (S[1], S[1])

1 + 0 + 0

i = 2, j = 1 + 2 + 255 = 2

i = 3, j = 2 + 3 + 254 = 3

⋮

i = 255, j = 254 + 255 + 2 = 255

i & j are same there state vector S will remain same.

P8.9

TRNG, each bit has same probability of being 0 or 1

bit not correlated

But bit stream is biased.

$P(b=1) = 0.5 + \partial$

$P(b=0) = 0.5 - \partial$ where $0 < \partial < 0.5$

(a)

Probability of Pairs, bits are independently chosen.

$P(00) = (0.5 - \partial)(0.5 - \partial) = (0.5 - \partial)^2$

$P(01) = (0.5 - \partial)(0.5 + \partial)$

$P(10) = (0.5 + \partial)(0.5 - \partial)$

$P(11) = (0.5 + \partial)(0.5 + \partial) = (0.5 + \partial)^2$

(b) The probability of 0 & 1 in modified sequence stands equal.

the probabability of occurrence $P(b=0) = P(b=1) = 0.5$

as Probability of 01 & 10 in the original stream are equal.

Chapter 9

P 9.4

$e = 65, \quad n = 2881$

$d = ?$

RSA.

1. p, q	
2. $n = pq$	$n = 2881 = p * q$
3. $\phi(n) = (p-1)(q-1)$	
4. $\gcd(e, \phi(n)) = 1$	$e = 65$
5. $d * e \equiv 1 \pmod{\phi(n)}$	$d * 65 \equiv 1 \pmod{}$

Trial & Error

$n = 2881 = p * q$.

we found $p = 43, \quad q = 67$

$\phi(n) = (p-1)(q-1) = 42 * 66 = 2772$

$d * 65 \equiv 1 \bmod 2772$

$d = \frac{2772 * Q + 1}{65}$

$Q = 17, \quad d = \frac{2772 * 17 + 1}{65} = \frac{47124 + 1}{65} = \frac{47125}{65}$

$d = 725.$

Using Extended Euclidean Algorithm.

$\Rightarrow$ 65 modulo $\phi(n)$

$e * d = 1 \pmod{\phi(n)}$

$65 * d = 1 \pmod{2772}$

$\gcd(a, b) = d = ax + by.$

$2772 * x + 65 * y = 1$ | $65d = 1 \bmod 2772$ | gcd(2772, 65

$P_i = P_{i-2} - P_{i-1} q_{i-2}$

Div		* Div	Rem	
2772	=	42 * (65)	+ 42	$P_0 = 0$
65	=	1 * (42)	+ 23	$P_1 = 1$
42	=	1 * (23)	+ 19	$P_2 = 0 - 1(42) = 2730$
23	=	1 * (19)	+ 4	$P_3 = 1 - 2730(42) = 1008$
19	=	4 * (4)	+ 3	$P_4 = 2730 - 1008(23) = 1722$
4	=	(1) * (3)	+ 1	$P_5 = 1008 - 1722(19) = 1554$
				$P_6 = 1722 - 1554(4) = 950$

65) 2772 (42
2730
42) 65 (1
42
23) 42 (1
23
19) 23 (1
19
4) 19 (4
16
3) 4 (1
3
1

Back substitution

$1 = 4 - 1(3)$

$1 = 4 - 1(19 - 4(4)) = 4 - 19 + 4(4)$

$= 4 - 19 + 4(23 - 1(19)) = 4 - 19 + 4 * 23 - 4(19)$

$= 4(1 + 23 - 19) - 19$

$= 4 - 19 + 4 * 23 - 4(42 - 23)$

$= 4 - 19 + 4 * 23 - 4 * 42 - 4 * 23$

$= 4(1 + 23 - 42 - 23) - 19$

$= -4(42) - 19$

$= -4(2772 - 42 * (65)) - (42 - 1(23))$

$= -4 * 2772 + 4 * 42 * 65 - 42 + 23$

$= -4 * 2772 + 168 * 65 - (2772 - 42(65)) + (65 - 42)$

$= -4 * 2772 + 168 * 65 - 2772 + 42(65) + 65 - 42$

$= -5(2772) + 211(65) - (2772 - 42(65))$

168
42
210

$= -6(2772) + 253(65)$

Back Substitution.

$-5(19) - 19$
$-19(5+1)$

$1 = 4 - 1(3)$

$= 4 - 1(19 - 4(4)) = 4 - 19 + 4(4) = 4(5) - 19$

$= 5(23 - 19) - 19 = 5(23) - 6(19)$

$= 5(23) - 6(42 - 23) = 5(23) - 6(42) + 6(23)$

$= 11(23) - 6(42) -$

$= 11(65 - 42) - 6(42) = 11(65) - 11(42) - 6(42) = 11(65) - 17(42)$

$= 11(65) - 17(2772 - 42(65))$

$11(65) - 17(2772) + 714(65)$

$-17(2772) + 725(65)$

$1 = -17(2772) + 725(65)$

47124 47125

$\gcd(2772, 65) = (d) = ax + by = (r_n)$ $[x_n = x_{n-2} - q_n x_{n-1}]$

i	q_i	r_i	x_i	y_i $[y_n = y_{n-2} - q_n y_{n-1}]$	
-1		2772	1	0	
0		65	0	1	
1	42	42	1	-42	$(0 - 1(42))$
2	1	23	-1	43	$(1 - (-42)(1))$
3	1	19	2	-85	$(-42 - (1)(43))$
4	1	4	-3	128	$(43 - 1(-85))$
5	4	3	14	-597	$(-85 - (128 \times 4))$
6	1	1	-17	725	(answer)

$-1 - 1(-1+1)$

$65 - 17 = 48$

$(65 * 725) \bmod 2772 = 1$

$(48 * 2772) \bmod 65 = 1$

Pg.8 RSA Algo

1. p, q $\quad n = p * q$ (very large n)
2. $\phi(n) = (p-1)(q-1)$
3. $\gcd(e, \phi(n)) = 1$
4. $de \equiv 1 \pmod{\phi(n)}$
5. $C = M^e \bmod n$ $\quad$ PU(n, e)
5. $M = C^d \bmod n$ $\quad$ PR(d, n)

A	0	$0^e \bmod n$
B	1	$1^e \bmod n$
C	2	$2^e \bmod n$
⋮	⋮	
Z	25	$25^e \bmod n$

Alphabet characters $\{A, B, \ldots, Z\}$

corresponding integers $\{0, 1, \ldots, 25\}$

corresponding ciphertext $\{0^e \bmod n, 1^e \bmod n, \ldots 25^e \bmod n\}$

Now, $\{n, e\}$ are public and therefore intruder can compute $m^e \bmod n$ for all all possible values of m (26 values)

Therefore, It is not secure, the most efficient attack against the scheme is to compute $m^e \bmod n$ for all possible values of M. ~~i.e. type of frequency attack~~. Then create a look-up table with ciphertext as an index and the corresponding plaintext as a value of the appropriate location in the table.

Frequency attack can also be applied here in this case, as we have only 26 possibilities.

Chapter 10

P10.1 Diffie-Hellman Key Exchange

Alice  Bob

prime number q, primitive root α

$\alpha < q$

generate private key

X_A, $X_A < q$

generate private key

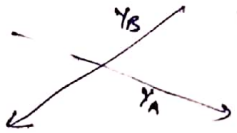
X_B, $X_B < q$

public key

$Y_A = \alpha^{X_A} \bmod q$

public key calculation

$Y_B = \alpha^{X_B} \bmod q$



$K = (Y_B)^{X_A} \bmod q$

$K = (Y_A)^{X_B} \bmod q$

Now, in question it is given that,

$q = 157$, $\alpha = 5$.

10.1 (b) $X_B = 27$ $\quad Y_B = (5)^{27} \bmod 157 = 65$ Ans

$\left[(5^2) * (5^5) * (5^7) * (5^{13})\right] \bmod 157$

$(25 * 142 * 96 * 22) \bmod 157 = 65$

$27 = 11011$ (32, 1, 0)

$5^1 \; 5^2 \; 5^8 \; 5^{16}$

3 | 27
3 | 9
3 | 3
1

P10.2 (b)

Given $q = 23$, $\alpha = 5$, $Y_A = 8$

$Y_A = \alpha^{X_A} \bmod q$, $8 = 5^{X_A} \bmod 23$

$X_A = d\log_{5,23}(8)$

$X_A = 6$

$8 = 5^{X_B} \bmod 23$

$y = g^x \bmod p$

$b = a^i \bmod p$

$d\log_{a,p} b$ num b for base a (mod p)

$d\log_{5,23}(8) = 6$

$Y_B = 10$, $Y_B = \alpha^{X_B} \bmod q$, $10 = 5^{X_B} \bmod 23$

$X_B = d\log_{5,23}(10) = 3$

$d\log_{13,19}(2) = 11$

$K = (Y_B)^{X_A} \bmod q = (10)^6 \bmod 23 = 6$

or $(Y_A)^{X_B} \bmod q = (8)^3 \bmod 23 = 6$ $\Big\}$ $K = 6$ Ans