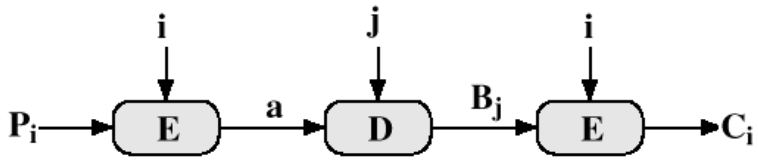
The University of Newcastle
School of Electrical Engineering and Computer Science

COMP3260/COMP6360 Data Security

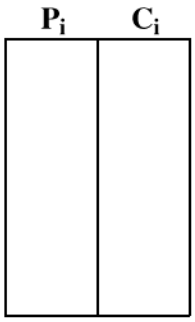
Week 7 Workshop – 11th and 12th April 2019
Solutions

1. Consider the known plaintext attack on triple DES with two keys as presented in the lectures. Show that the expected effort for this attack is of order $2^{120-\lg n}$, where n is the number of plaintext-ciphertext pairs available to the intruder.

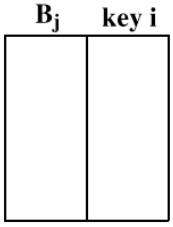
Solution: The following known-plaintext attack on triple DES, due to Oorschot and Wiener, 1990, was considered in the lecture.



1) Obtain n known plaintext-ciphertext pairs (P, C). Place them in a table sorted on the values of P.



2) Pick an arbitrary value a. For each of the possible 2^{56} keys K_1 = i, calculate the plaintext value P_i that produces a: $P_i = D_i$ [a].
3) For each P_i that matches an entry in the first table create an entry in the second table consisting of the K_1 and $B = D_i$ [C]



4) For each of the possible 2^{56} possible keys K_2 = j, calculate $B_j = D_j$ [a] and check if there is a match in table 2, in which case (i,j) is a candidate pair of keys.
5) Test each candidate pair of keys on a few plaintext-ciphertext pairs; if no pair succeeds, repeat from step 1 with a new value of a.

If we only have one plaintext-ciphertext pair, the probability that we selected the correct *a* is $p=1/2^{64}$. For n plaintext-ciphertext pairs, the probability that we selected the correct *a* is $n/2^{64}$.

We now need to use a result from probability theory, which states that if we have a bag with *N* balls, and if *n* of them are red, then the expected number of times we need to draw a ball without a replacement to draw one red ball is $(N+1)/(n+1)$. In our case this means that the expected number of *a*'s we need to try is $(2^{64}+1)/(n+1)$ and for large *n* we have $(2^{64}+1)/(n+1) \approx 2^{64}/n$. Since for each *a* we need to try 2^{56} keys (first for $K_1=i$ and then for $K_2=j$), the expected effort for the attack is $2^{56}2^{64}/n=2^{120-\lg n}$.

Note that in the above derivation we ignored the effort of searching through n plaintexts.

2. Find the unicity distance of Triple-DES, and the Advanced Encryption Standard that uses the key which uses key with 128, 192 or 256 bits

Solution:

Triple-Des has key length 168 bits, so it has 2^{168} possible keys:
$H(K) = \log_2 2^{168} = 168$ Bits
$N = H(K)/D = 168/3.2 = 52.5$ bits

AES 128 bit key: $H(K) = \log_2 2^{128} = 128$ Bits; $N = H(K)/D = 128/3.2 = 40$ bits

AES 192 bit key: $H(K) = \log_2 2^{192} = 192$ Bits; $N = H(K)/D = 192/3.2 = 60$ bits

AES 256 bit key: $H(K) = \log_2 2^{256} = 256$ Bits; $N = H(K)/D = 256/3.2 = 80$ bits

3. The first stage in each round of the AES is Substitute Bytes Transformation which uses 16×16 S-box. Bytes of the State are replaced one at the time, in the following way: the leftmost 4 bits of a byte determine the row and the rightmost 4 bits determine the column in the S box. The first row and the first column of the S-box are shown below:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA															
2	B7															
3	04															
4	09															
5	53															
6	D0															
7	51															
8	CD															
9	60					2A										
A	E0															

B	E7															
C	BA															
D	70															
E	E1															
F	8C															

The S-box is constructed in the following way:

a. The S-box is initialized with byte values where leftmost 4 bits correspond to the row and the rightmost 4 bits correspond to columns, that is, the value of the byte in row *x* and column *y* is *xy*.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
1	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
2	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
3	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
4	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
5	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
6	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
7	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
8	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
9	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

b. Each byte is mapped into its multiplicative inverse in GF(2^8) with irreducible polynomial $p(x) = x^8 + x^4 + x^3 + x + 1$; byte 00 is mapped into itself.

c. The following transformation is applied to each byte $(b_7,b_6,b_5,b_4,b_3,b_2,b_1,b_0)$, where the values of *c* is $(c_7,c_6,c_5,c_4,c_3,c_2,c_1,c_0) = (01100011)$:
$b_i' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$, or, in matrix form **B'** = **XB** ⊕ **C**:

$$\begin{Vmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{Vmatrix} = \begin{Vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{Vmatrix} \begin{Vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{Vmatrix} \oplus \begin{Vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{Vmatrix}$$

Show that the byte in row labeled 9 and column labeled 5 has value 2A. (Hint: Multiplicative inverse of 95 in GF(2^8) with irreducible polynomial $p(x) = x^8 + x^4 + x^3 + x + 1$ is 8A; try to find this yourself!)

Solution : See text.

4. The Inverse Substitute Byte Transformation uses the inverse S-box, which is constructed by first applying the inverse of the transformation B' = XB ⊕ C (we denote this transformation as B' = YB ⊕ D), and then taking the multiplicative inverse in GF(28) with irreducible polynomial $p(x) = x^8 + x^4 + x^3 + x + 1$. The inverse transformation is

$b_i' = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$

In matrix form we have

$$\begin{Vmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{Vmatrix} = \begin{Vmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{Vmatrix} \begin{Vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{Vmatrix} \oplus \begin{Vmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix}$$

Prove that inverse S-box is indeed the inverse of S-box.

Solution :

We need to show that **B = YB' ⊕ D = Y (XB ⊕ C) ⊕ D**

$$\begin{Vmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{Vmatrix} \begin{Vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{Vmatrix} \begin{Vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{Vmatrix} \oplus \begin{Vmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{Vmatrix} \begin{Vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{Vmatrix} \oplus \begin{Vmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} =$$

$$\begin{Vmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{Vmatrix} \begin{Vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{Vmatrix} \oplus \begin{Vmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \oplus \begin{Vmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} = \begin{Vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{Vmatrix}$$

5. a. What is $\{53\}^{-1}$ in GF(2^8)?
b. Verify the entry for {53} in the S-box.

Solution:

a. {01}
b. We have
c.

$$\begin{Vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{Vmatrix} \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} \oplus \begin{Vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{Vmatrix} = \begin{Vmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \oplus \begin{Vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{Vmatrix}$$

In hexadecimal notation, this is {7C}, which is indeed the value in row 0, column 1 in the S-box.

6. Compare AES to DES. For each of the following elements of DES, indicate the comparable element in AES or explain why it is not needed in AES.

- ***a.*** XOR of subkey with the input to the f function.
- ***b.*** XOR of the f function output with the left half of the block
- ***c.*** f function
- ***d.*** permutation P
- ***e.*** swapping of halves of the block

Solution:

a. AddRoundKey
b. This best corresponds to the MixColumns, as there different bit bits affect each other.
c. F function contains S boxes, which are the non-linear elements; therefore, it corresponds to SubstituteBytes
d. ShiftRows, which permutes the bytes
f. There is no exact match for swapping of halves of the block; as it permutes the bits, it is most related to ShiftRows.