Suggested Solutions to Homework Assignment #1B

1. Exercise problems from [Stallings 2011]:

5.1 We want to show that $d(x) = a(x) \times b(x) \bmod (x^4 + 1) = 1$. Substituting into Equation (5.12) in Appendix 5A, we have:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} 0\text{E} \\ 09 \\ 0\text{D} \\ 0\text{B} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

But this is the same set of equations discussed in the subsection on the MixColumn transformation:

$$\begin{aligned} &(\{0\text{E}\} \cdot \{02\}) \oplus \{0\text{B}\} \oplus \{0\text{D}\} \oplus (\{09\} \cdot \{03\}) = \{01\} \\ &(\{09\} \cdot \{02\}) \oplus \{0\text{E}\} \oplus \{0\text{B}\} \oplus (\{0\text{D}\} \cdot \{03\}) = \{00\} \\ &(\{0\text{D}\} \cdot \{02\}) \oplus \{09\} \oplus \{0\text{E}\} \oplus (\{0\text{B}\} \cdot \{03\}) = \{00\} \\ &(\{0\text{B}\} \cdot \{02\}) \oplus \{0\text{D}\} \oplus \{09\} \oplus (\{0\text{E}\} \cdot \{03\}) = \{00\} \end{aligned}$$

The first equation is verified in the text. For the second equation, we have $\{09\} \cdot \{02\} = 00010010$; and $\{0\text{D}\} \cdot \{03\} = \{0\text{D}\} \oplus (\{0\text{D}\} \cdot \{02\}) = 00001101 \oplus 00011010 = 00010111$. Then

$$\begin{array}{lcl} \{09\} \cdot \{02\} & = & 00010010 \\ \{0\text{E}\} & = & 00001110 \\ \{0\text{B}\} & = & 00001011 \\ \{0\text{D}\} \cdot \{03\} & = & \underline{00010111} \\ & & 00000000 \end{array}$$

For the third equation, we have $\{0\text{D}\} \cdot \{02\} = 00011010$; and $\{0\text{B}\} \cdot \{03\} = \{0\text{B}\} \oplus (\{0\text{B}\} \cdot \{02\}) = 00001011 \oplus 00010110 = 00011101$. Then

$$\begin{array}{lcl} \{0\text{D}\} \cdot \{02\} & = & 00011010 \\ \{09\} & = & 00001001 \\ \{0\text{E}\} & = & 00001110 \\ \{0\text{B}\} \cdot \{03\} & = & \underline{00011101} \\ & & 00000000 \end{array}$$

For the fourth equation, we have $\{0\text{B}\} \cdot \{02\} = 00010110$; and $\{0\text{E}\} \cdot \{03\} = \{0\text{E}\} \oplus$

$(\{0E\} \cdot \{02\}) = 00001110 \oplus 00011100 = 00010010$. Then

$$\begin{array}{lcl} \{0B\} \cdot \{02\} & = & 00010110 \\ \{0D\} & = & 00001101 \\ \{09\} & = & 00001001 \\ \{0E\} \cdot \{03\} & = & \underline{00010010} \\ & & 00000000 \end{array}$$

Thus, we found out $d(x) = a(x) \times b(x) \bmod (x^4 + 1) = 1$ by calculating these four equations.

5.4 **a.**

00	04	08	0C
01	05	09	0D
02	06	0A	0E
03	07	0B	0F

b.

01	05	09	0D
00	04	08	0C
03	07	0B	0F
02	06	0A	0E

c.

7C	6B	01	D7
63	F2	30	FE
7B	C5	2B	76
77	6F	67	AB

d.

7C	6B	01	D7
F2	30	FE	63
2B	76	7B	C5
AB	77	6F	67

e.

75	87	0F	B2
55	E6	04	22
3E	2E	B8	8C
10	15	58	0A

5.6 **a.** AddRoundKey

b. The MixColumn step, because this is where the different bytes interact with each other.

c. The ByteSub step, because it contributes nonlinearity to AES.

d. The ShiftRow step, because it permutes the bytes.

e. There is no wholesale swapping of rows or columns. AES does not require this step because: The MixColumn step causes every byte in a column to alter every other byte in the column, so there is not need to swap rows; The ShiftRow step moves bytes from one column to another, so there is no need to swap columns

6.4 **a.** The question assumes that there was an error in block C_4 of the transmitted ciphertext.

ECB mode: In this mode, ciphertext block C_i is used only as input for the direct dencryption of plaintext block P_i. Therefore, a transmission error in block C_4 will only corrupt block P_4 of the decrypted plaintext.

CBC mode: In this mode, ciphertext block C_i is used as input to the XOR function when obtaining plaintext blocks P_i and P_{i+1}. Therefore, a transmission error in block C_4 will corrupt blocks P_4 and P_5 of the decrypted plaintext, but will not propagate to any of the other blocks.

CTR mode: In this mode, ciphertext block C_i, as well as the encrypted counter ti, are used only as input for the direct decryption of plaintext block P_i. Therefore, a transmission error in block C_4 will only corrupt block P_4 of the decrypted plaintext.

b. The question assumes that the ciphertext contains N blocks, and that there was a bit error in the source version of P_3.

ECB mode: In this mode, ciphertext block C_i is generated by direct encryption of plaintext block P_i, independent of the other plaintext or ciphertext blocks. Therefore, a bit error in block P_3 will only affect ciphertext block C_3 and will not propagate further. Thus, only one ciphertext block will be corrupted.

CBC mode: In this mode, ciphertext block C_i is generated by XORing plaintext block Pi with ciphertext block C_{i-1}. Therefore, a bit error in block P_3 will affect ciphertext block C_3, which in turn will affect ciphertext block C_4 and so forth, and therefore the error will propagate through all remaining ciphertext blocks. Thus, $N-2$ ciphertext block will be corrupted.

CTR mode: In this mode, ciphertext block C_i is generated by applying the XOR function to plaintext block P_i and the encrypted counter t_i, independent of the other plaintext or ciphertext blocks. Therefore, a bit error in block P_3 will only affect ciphertext block P_3 and will not propagate further. Thus, only one ciphertext block will be corrupted.

6.7 For this padding method, the padding bits can be removed unambiguously, provided the receiver can determine that the message is indeed padded. One way to ensure that the receiver does not mistakenly remove bits from an unpadded message is to require the sender to pad every message, including messages in which the final block is already complete. For such messages, an entire block of padding is appended.

6.11 **a.** CTS is the same as CBC, except for the last two blocks. P_{N-1} is encrypted as usual for CBC, but the result of this encryption is split into two parts: the prefix is used as C_N, while the remaining bits (X) are used in the encryption of P_N into C_{N-1} but are not returned. Since the X bits are used in future encryption stages they can be retrieved during decryption, but the ciphertext remains the same length as the original plaintext (unlike with simple padding schemes). This is useful when we wish the ciphertext to fit in the same buffer as the plaintext did.

b. Decrypting C_{N-1}: Assume the message length is (block_size*i - j). After passing C_{N-1} through the encryption/decryption box using K, denote the result as Z. We XOR the (block_size-j) prefix bits of Z with C_N, and that gives us P_N.

Decrypting C_N: After the above procedure, concatenate the j postfix bits of Z at the end of C_N, and pass these through the encryption/decryption box using K. The result is XORed with C_{N-2} and P_{N-1} is the result.

c. The specific value of the padding of P_N is not important, as long as the entity decrypting the message knows what the padding is. Thus 1's or a key prefix would not obstruct decryption. Using 0's is just the simplest option since it leaves the X bits as they were.

6.12 **a.** For all blocks other than the last, this is simply CBC. We therefore focus on the last block.

Encryption: Straightforward from the diagram. C_{N-1} is re-encrypted with the key K, and the j leftmost bits are XORed with PN to produce CN.

Decryption: C_{N-1} is re-encrypted with the key K, and the j leftmost bits are XORed with C_N to produce P_N.

b. The property does not hold. Specifically, for the last block we need to use encryption of C_{N-1}, a part of the ciphertext, in order to decrypt part of the ciphertext. For a standard symmetric cipher (e.g. CBC), only decryption of ciphertext blocks would be used during decryption.

c. TS is better, since the property mentioned in section (b) holds for it, making implementation simpler.

2. In the CTR mode, the seed value (V) will be incremented by 1 after each encryption. Thanks to the invertibility of the encryption algorithm, different values of V give rise to different pseudorandom bits. Only when the value of V loops back to the initial value, the whole stream will repeat. V has 2^{128} possible values, each producing 128 pseudorandom bits. So, the period of the pseudorandom bit stream is 128×2^{128} bits long.

3. To compute the expected period of the bit stream, we have to compute the expected number of times\rounds the encryption algorithm has to be applied to get a repeated stream. Multiplying this expected value with the block length, we get the expected period of the bit stream.

In the following table, the first column is the number of rounds that the encryption algorithm have been applied. The third column is the probability that after these many rounds the generated bit stream repeats. And the initial seed value is V_0.

Let us look at the second row of Column 3. The first part $\frac{1}{2^{128}-1}$ is the probability that $V_0 = V_2$, given that $V_0 \neq V_1$. We know that the AES encryption algorithm is invertible. When $V_0 \neq V_1$, it is not possible that the new generated block value (V_2) is equal to last round's block value (V_1). So the number of possible values of this generated block is $2^{128} - 1$. The second part is the computation about the probability that $V_0 \neq V_1$.

rounds	seed values transition	probability that the generated bit stream repeat
1	$V_0 \rightarrow V_1$	$\frac{1}{2^{128}}$
2	$V_0 \rightarrow V_1 \rightarrow V_2$	$\frac{1}{2^{128}-1} \times \frac{2^{128}.(2^{128}-1)}{(2^{128})^2} = \frac{1}{2^{128}}$
3	$V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3$	$\frac{1}{2^{128}-2} \times \frac{2^{128}.(2^{128}-1)(2^{128}-2)}{2^{128}.2^{128}.(2^{128}-1)} = \frac{1}{2^{128}}$
$\vdots$	$\vdots$	$\vdots$
2^{128}	$V_0 \rightarrow V_1 \rightarrow \cdots \rightarrow V_{2^{128}}$	$\frac{1}{2^{128}}$

The expected number of rounds to get a repeated bit stream:

$$1 \times \frac{1}{2^{128}} + 2 \times \frac{1}{2^{128}} + \cdots + 2^{128} \times \frac{1}{2^{128}}$$
$$= (1 + 2 + \cdots + 2^{128}) \times \frac{1}{2^{128}}$$
$$= \frac{(2^{128}+1)\times 2^{128}}{2} \times \frac{1}{2^{128}}$$
$$= \frac{(2^{128}+1)}{2}$$

So, the expected period is $128 \times \frac{(2^{128}+1)}{2} = 64 \times (2^{128} + 1)$ bits long.