C&O 487/687: Assignment #2 Solutions

Due date: October 10, 2024 (Thursday)

Note: Some questions use randomization to customize to you specifically. Please include your max-8-character UW user id (`b54khan`) at the beginning of your answer so we can look up your custom solution.

1. [10 marks] **Linear cryptanalysis**

   Please include your max-8-character UW user id (`b54khan`) at the beginning of your answer so we can look up your custom solution.

   This problem uses the simplified cipher described in Section 2 of "A Tutorial on Linear and Differential Cryptanalysis" by Howard M. Heys, available at
   `http://www.engr.mun.ca/~howard/Research/Papers/ldc_tutorial.html`
   We refer to this cipher as the "Heys cipher".

   For the purposes of this problem, each student has a fixed, unknown 80-bit key. You will be carrying out a known-plaintext attack against the Heys cipher using linear cryptanalysis, using a set of 20,000 distinct random plaintext-ciphertext pairs. You can download your plaintexts and ciphertexts (unique to you) at the following addresses:
   `https://www.math.uwaterloo.ca/~dstebila/as/as.cgi/download/co487_f24/a2q1plaintexts.txt`
   `https://www.math.uwaterloo.ca/~dstebila/as/as.cgi/download/co487_f24/a2q1ciphertexts.txt`

   The format of the files is that the nth line of the ciphertext file equals the encryption of the nth line of the plaintext file under your secret key.

   For the programming aspects of questions 1.a.i, 1.b, and 1.d.ii, you may work with a partner to do the programming and you may submit the same computer program source code, but you must submit your own write-up and explanations for the non-programming parts of those questions. Please indicate in your submission who you worked with.

   Solution. Your key is:
   0000010110111011001001010111111100110111011101101010101001011000101011001101 1000

   (a) [2 marks] Using the linear approximation $U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 \cong 0$, Carol guesses that the target partial subkey $[K_{5,5}, K_{5,6}, K_{5,7}, K_{5,8}, K_{5,13}, K_{5,14}, K_{5,15}, K_{5,16}]$ has the value $[0,1,1,1,0,1,1,0]$. Note that Carol's guess is not necessarily correct! Do one of the following:

   i. Determine the magnitude of the bias for this partial subkey value over your twenty thousand plaintext/ciphertext pairs, using a computer program, and provide the source code for your program, or:

   ii. Determine the magnitude of the bias for this partial subkey value over your first *ten* plaintext/ciphertext pairs, without using a computer program, and show your work.

   Solution. Using the partial subkey guess

   $$[K_{5,5}, K_{5,6}, K_{5,7}, K_{5,8}, K_{5,13}, K_{5,14}, K_{5,15}, K_{5,16}] = [0,1,1,1,0,1,1,0],$$

   we compute:

   i. The equation $U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0$ holds for exactly 10101 of the 20000 given plaintext/ciphertext pairs. The bias of this approximation is

   $$\frac{10101}{20000} - \frac{1}{2} = \frac{101}{20000} = +0.00505$$

   and its magnitude is $|+0.00505| = 0.00505$.

ii. We compute by hand:

Ciphertext 1			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		0010	1010
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
0101	1100	1100	1011
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 1$			

Ciphertext 2			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		1001	1110
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
1110	1000	0000	0111
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0$			

Ciphertext 3			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		1010	1101
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
1101	1011	0010	0110
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 1$			

Ciphertext 4			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		1111	1010
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
1000	1100	0111	1011
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 1$			

Ciphertext 5			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		1100	0101
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
1011	0011	0110	1000
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 1$			

Ciphertext 6			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		0001	0011
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
0110	0101	1010	1100
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 1$			

Ciphertext 7			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		0001	1000
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
0110	1110	1010	0000
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0$			

Ciphertext 8			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		1110	1010
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
1001	1100	1101	1011
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 1$			
Ciphertext 9			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		1001	1111
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
1110	1001	0000	1101
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0$			
Ciphertext 10			
$P_5P_6P_7P_8$		$C_5C_6C_7C_8$	$C_{13}C_{14}C_{15}C_{16}$
0000		1110	0001
$V_{4,5}V_{4,6}V_{4,7}V_{4,8}$	$V_{4,13}V_{4,14}V_{4,15}V_{4,16}$	$U_{4,5}U_{4,6}U_{4,7}U_{4,8}$	$U_{4,13}U_{4,14}U_{4,15}U_{4,16}$
1001	0111	1101	1111
$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0$			

The equation $U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0$ holds for exactly 5 of the 10 given plaintext/ciphertext pairs. The bias is

$$\frac{5}{10} - \frac{1}{2} = \frac{0}{10} = +0$$

and its magnitude is 0.

(b) [3 marks] Find the value of the partial subkey $[K_{5,5}, K_{5,6}, K_{5,7}, K_{5,8}, K_{5,13}, K_{5,14}, K_{5,15}, K_{5,16}]$ for your key, by calculating the target partial subkey which yields the largest magnitude of bias over your 20,000 plaintext/ciphertext pairs. You will almost certainly need a computer program for this task; provide the source listing for any computer code that you or your collaborators write.

Solution. For each possible value of the partial subkey

$$[K_{5,5}, K_{5,6}, K_{5,7}, K_{5,8}; K_{5,13}, K_{5,14}, K_{5,15}, K_{5,16}]$$

we compute the value of U_4 for each of our given ciphertexts, under the assumption that the subkey equals this value. Using these U_4 values, we compute bias of the approximation

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 \approx 0$$

over the 20000 given plaintext/ciphertext pairs. The partial subkey value exhibiting the largest bias magnitude (regardless of sign) is likely to be the correct subkey.

Using a computer program, we find via brute-force search over all 256 possibilities that the values of the subkey bits exhibiting the largest bias magnitudes (sorted by bias magnitude) are:

Key bits 5-8	Key bits 13-16	Number of zeros	Bias
1100	1000	9356	−0.032200
1101	1000	9527	−0.023650
1111	1000	10462	+0.023100
1100	1010	10447	+0.022350
1101	1010	10446	+0.022300
0001	1000	10387	+0.019350
1110	1000	10387	+0.019350
1100	0110	9616	−0.019200
1111	1010	9643	−0.017850
0001	1010	9680	−0.016000
0000	1000	10312	+0.015600
0001	0111	9690	−0.015500
1100	1011	10305	+0.015250
1110	0110	10305	+0.015250
0010	1000	9699	−0.015050
1111	0100	9704	−0.014800
0100	0011	10269	+0.013450
1010	1010	10256	+0.012800
1011	1111	10250	+0.012500
0011	0111	10249	+0.012450

From this calculation, we conclude that the subkey bits are $1100, 1000$.

(c) [2 marks] By using Table 4 in the tutorial, compute the bias in each of the following individual S-box linear approximations:

$$S_{11} : X_1 \oplus X_4 \cong Y_1$$

$$S_{13} : X_1 \oplus X_4 \cong Y_1$$

$$S_{21} : X_1 \oplus X_3 \cong Y_2$$

$$S_{32} : X_1 \cong Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4$$

Then, combine these to find a linear approximation of the first three rounds of the Heys cipher, and calculate the theoretical magnitude of its bias.

Solution. From the approximation for S_{11}, we derive the approximation

$$V_{1,1} \approx U_{1,1} \oplus U_{1,4} = P_1 \oplus P_4 \oplus K_{1,1} \oplus K_{1,4}$$

with bias $-4/16 = -1/4$.

From the approximation for S_{13}, we derive the approximation

$$V_{1,9} \approx U_{1,9} \oplus U_{1,12} = P_9 \oplus P_{12} \oplus K_{1,9} \oplus K_{1,12}$$

with bias $-4/16 = -1/4$.

From the approximation for S_{21}, we derive the approximation

$$V_{2,2} \approx U_{2,1} \oplus U_{2,3} = V_{1,1} \oplus V_{1,9} \oplus K_{2,1} \oplus K_{2,3}$$

with bias $-4/16 = -1/4$.

From the approximation for S_{32}, we derive the approximation

$$V_{3,5} \oplus V_{3,6} \oplus V_{3,7} \oplus V_{3,8} \approx U_{3,5} = V_{2,2} \oplus K_{3,5}$$

with bias $-6/16 = -3/8$.

Combining all the approximations, we obtain

$$\begin{aligned} V_{3,5} \oplus V_{3,6} \oplus V_{3,7} \oplus V_{3,8} &\approx U_{3,5} \\ &= V_{2,2} \oplus K_{3,5} \\ &\approx U_{2,1} \oplus U_{2,3} \oplus K_{3,5} \\ &= V_{1,1} \oplus V_{1,9} \oplus K_{2,1} \oplus K_{2,3} \oplus K_{3,5} \\ &\approx (P_1 \oplus P_4 \oplus K_{1,1} \oplus K_{1,4}) \oplus (P_9 \oplus P_{12} \oplus K_{1,9} \oplus K_{1,12}) \\ &\quad \oplus K_{2,1} \oplus K_{2,3} \oplus K_{3,5} \end{aligned}$$

with bias

$$2^3 \cdot \left(-\frac{1}{4}\right) \cdot \left(-\frac{1}{4}\right) \cdot \left(-\frac{1}{4}\right) \cdot \left(-\frac{3}{8}\right) = +\frac{3}{64}.$$

On the other hand,

$$V_{3,5} \oplus V_{3,6} \oplus V_{3,7} \oplus V_{3,8} = U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus K_{4,2} \oplus K_{4,6} \oplus K_{4,10} \oplus K_{4,14}$$

(with 100% certainty). We conclude that the linear approximation

$$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} \approx 0$$

holds with bias $\pm 3/64$, with the sign of the bias depending on the value of the key bits.

(d) [3 marks] Do *one* of the following:

i. Using the linear approximation from part (c), determine the entire subkey K_5. You will almost certainly need a computer program for this task; provide the source listing for any computer code that you or your collaborators write, or:

ii. Using the (incorrect) guess $K_5 = [1100011111100110]$ for the fifth subkey, determine the magnitude of the bias of the approximation from part (c) for this subkey over your first ten plaintext/ciphertext pairs, without using a computer program, and show your work.

Solution.

i. For each possible value of the partial subkey

$$\begin{aligned} &[K_{5,1}, K_{5,2}, K_{5,3}, K_{5,4}, K_{5,5}, K_{5,6}, K_{5,7}, K_{5,8}, \\ &K_{5,9}, K_{5,10}, K_{5,11}, K_{5,12}, K_{5,13}, K_{5,14}, K_{5,15}, K_{5,16}] \end{aligned}$$

we compute the value of U_4 for each of our given ciphertexts, under the assumption that K_5 equals this value. Using these U_4 values, we compute bias of the approximation

$$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 0$$

over the 10000 given plaintext/ciphertext pairs. The partial subkey value exhibiting the largest bias magnitude (regardless of sign) is likely to be the correct subkey. Note that we already know the values of $K_{5,5}, K_{5,6}, K_{5,7}, K_{5,8}$ and $K_{5,13}, K_{5,14}, K_{5,15}, K_{5,16}$ from the previous part, so we only need to guess the remaining eight bits.
Using a computer program, we find via brute-force search over all 256 possibilities that the values of the subkey bits

$$[K_{5,1}, K_{5,2}, K_{5,3}, K_{5,4}, K_{5,9}, K_{5,10}, K_{5,11}, K_{5,12}]$$

exhibiting the largest bias magnitudes (sorted by bias magnitude) are:

Key bits 1-4	Key bits 9-12	Number of zeros	Bias
1010	1101	10895	+0.044750
1000	1101	10556	+0.027800
1100	1101	9445	−0.027750
1110	1101	9478	−0.026100
1010	1001	9484	−0.025800
1010	1010	10515	+0.025750
1011	1101	9508	−0.024600
1010	1011	9524	−0.023800
1001	1101	9541	−0.022950
1101	1101	10458	+0.022900
1010	1110	9576	−0.021200
1010	1111	10385	+0.019250
1010	1100	9616	−0.019200
1000	1110	9647	−0.017650
1011	1010	9654	−0.017300
1011	1100	10307	+0.015350
1100	1010	9699	−0.015050
1001	1001	10298	+0.014900
1100	1100	10294	+0.014700
1110	1001	10293	+0.014650

From this calculation, we conclude that $K_5 = 1010\ 1100\ 1101\ 1000$.

ii. We compute by hand:

Ciphertext 1	
P	C
0000000000000001	0111001010011010
V_4	U_4
1011 0101 0111 1100	0110 1100 1111 1011
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 1$	
Ciphertext 2	
P	C
0000000000000010	1111100111001110
V_4	U_4
0011 1110 0010 1000	1000 0000 0100 0111
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 0$	
Ciphertext 3	
P	C
0000000000000011	0111101011001101
V_4	U_4
1011 1101 0010 1011	0110 0010 0100 0110
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 1$	
Ciphertext 4	
P	C
0000000000000100	1001111111001010
V_4	U_4
0101 1000 0010 1100	1100 0111 0100 1011
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 1$	

Ciphertext 5	
P	C
0000000000000110	1000110011000101
V_4	U_4
0100 1011 0010 0011	0001 0110 0100 1000
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 0$	
Ciphertext 6	
P	C
0000000000001000	0100000100010011
V_4	U_4
1000 0110 1111 0101	0111 1010 0101 1100
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 1$	
Ciphertext 7	
P	C
0000000000001010	0111000101111000
V_4	U_4
1011 0110 1001 1110	0110 1010 1101 0000
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 0$	
Ciphertext 8	
P	C
0000000000010001	0001111011111010
V_4	U_4
1101 1001 0001 1100	0010 1101 0011 1011
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 0$	
Ciphertext 9	
P	C
0000000000010101	0011100111011111
V_4	U_4
1111 1110 0011 1001	0101 0000 1000 1101
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 1$	
Ciphertext 10	
P	C
0000000000011010	0010111010000001
V_4	U_4
1110 1001 0110 0111	0000 1101 1010 1111
$U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 1$	

The equation $U_{4,2} \oplus U_{4,6} \oplus U_{4,10} \oplus U_{4,14} \oplus P_1 \oplus P_4 \oplus P_9 \oplus P_{12} = 0$ holds for exactly 4 of the 10 given plaintext/ciphertext pairs. The bias is

$$\frac{4}{10} - \frac{1}{2} = -\frac{1}{10} = -0.1$$

and its magnitude is 0.1.

(e) [2 bonus marks] A small amount of extra credit is available if you can determine any additional key bits.

2. [12 marks] **Penguin or not a penguin? (Block cipher modes of operation)**

This question is meant to illustrate the importance of proper implementation. The Jupyter notebook `Penguins.ipynb` (available on LEARN) contains Python implementations of four different block

cipher modes of operation: ECB, CBC, OFB, and CTR. Some have been implemented correctly, and some have not. You have also been provided with four encrypted image files, which you can download from LEARN:

- `image1.ppm`, which was encrypted using ECB mode
- `image2.ppm`, which was encrypted using CBC mode and the IV specified in the Jupyter notebook
- `image3.ppm`, which was encrypted using OFB mode and the IV specified in the Jupyter notebook
- `image4.ppm`, which was encrypted using CTR mode and the IV specified in the Jupyter notebook

Some of the above files are the encryptions of images of penguins, and some are not. By analyzing the encrypted images and the code in `Penguins.ipynb` that was used to encrypt them, your task is to determine which of the images are penguins, and which are not.

For each image/mode, you should explain:

- Whether there is a flaw in the implementation of the mode of operation, and what the flaw is.
- Whether the plaintext image is:
  (a) DEFINITELY A PENGUIN
  (b) DEFINITELY NOT A PENGUIN
  (c) CANNOT BE CERTAIN WHETHER OR NOT IT IS A PENGUIN

In each case, you must fully justify your answer. (Note: "Because it looks like a penguin" is not a fully justified answer. However this doesn't mean you have to fully decrypt every image.)

Solution.

- The image encrypted using ECB mode is definitely not a penguin. It is either a banana, a pumpkin, a palm tree, or a Christmas tree. In ECB mode, we have that if $m_1 = m_2$, then $E_k(m_1) = E_k(m_2)$. So, a collection of pixels which all have the same colour in the ciphertext image must also all have the same colour as each other in the original image. So, it is not possible under any key for a penguin to encrypt to something resembling a banana/pumpkin/palm tree/Christmas tree.
- The image encrypted using CBC mode is not a penguin. It is a goose. The `CBC_mode` function was not implemented properly: the line to call the block cipher should have been `AES_one_block(key, x)`, but instead it is written as `AES_one_block(iv+iv, x)`. Since the IV is public, the following function can be used to decrypt ciphertexts that were encrypted in this faulty way:

```
def CBC_mode_decrypt(iv, ct):
    pt = b''
    for i in range(0, len(ct), 16):
        inputblock = ct[i:i+16]
        dec = AES_decrypt_one_block(iv+iv, inputblock)
        pt += bytes(dec[j] ^ iv[j] for j in range(0, 16))
        iv = inputblock
    return pt
```

- There is no way of knowing whether the image encrypted using OFB mode is a penguin. The `OFB_mode` function was not implemented properly: rather than XORing the stream with the message blocks, the stream is XORed with the IV, resulting in a ciphertext that is dependent only on the key and the IV (in particular, said ciphertext is completely independent of the message). So, not even someone who knows the secret key would be able to tell whether the given image is the encryption of a penguin or not.

- The image encrypted using CTR mode is definitely a penguin. The `CTR_mode` function was not implemented properly: the counter is never incremented. So, it is essentially just ECB mode again, but with an extra XOR. This does not change the property that if $m_1 = m_2$, then $E_k(m_1) = E_k(m_2)$. So, a collection of pixels which all have the same colour in the ciphertext image must also all have the same colour as each other in the original image. Hence, if the ciphertext image resembles a penguin, then so must the original image.

3. [3 marks] **OFB mode**

   We generally assume that the adversary does not know the secret key, but does know the initialization vector. Suppose that, for OFB mode, we kept the IV secret as well. Show that this does not make an exhaustive key search any more expensive – in other words, show how to perform a brute-force attack on OFB with an unknown key and an unknown IV. What requirements have you made on the plaintext and ciphertext?

   Solution. Suppose we have a single plaintext-ciphertext pair that is at least 3 blocks in length: $(p_1 \| p_2 \| p_3, c_1 \| c_2 \| c_3)$.

   By the definition of OFB mode, these values satisfy:

   - $c_1 = p_1 \oplus O_1$ where $O_1 = E_k(iv)$
   - $c_2 = p_2 \oplus O_2$ where $O_2 = E_k(O_1)$
   - $c_3 = p_3 \oplus O_3$ where $O_3 = E_k(O_2)$

   Given p and c, but not k and iv, we can compute $O_1 = p_1 \oplus c_1$ and $O_2 = p_2 \oplus c_2$. For each possible k', check if $E_{k'}(O_1) = O_2$. If yes, then k' is a candidate for k. For all candidates k' we identify, check if $E_{k'}(O_2) = O_3$; this will narrow the field much further. If the message block size is at least 128 bits, then this will uniquely identify the correct key with overwhelming probability.

4. [7 marks] **Hash function properties**

   Trits are similar to bits but have three possible values $\{0, 1, 2\}$ instead of two. As such, they can be seen as elements in $\mathbb{Z}_3$ – integers modulo 3. They are sometimes used in algorithms where the objects we are working with have three possible states, such as walks on 3-regular graphs.

   For two strings of trits with same length $(t_1, \ldots, t_n)$ and $(t'_1, \ldots, t'_n)$, we can define the subtraction operation as:

   $$(t_1, \ldots, t_n) - (t'_1, \ldots, t'_n) := (t_1 - t'_1, \ldots, t_n - t'_n)$$

   where each tritwise operation is done modulo 3.

   Let $f : \{0, 1, 2\}^m \to \{0, 1, 2\}^m$ be a preimage-resistant bijection.

   For $x \in \{0, 1, 2\}^{2m}$ write $x = x' \| x''$ (in other words, split the $2m$-trit string x into two m-trit halves x' and x''). Define a new hash function $H : \{0, 1, 2\}^{2m} \to \{0, 1, 2\}^m$ as

   $$H(x) = f(x' - x'')$$

   (a) [6 marks] For sufficiently large values of m (e.g., $m = 256$), does H have each of the following desired properties for a cryptographic hash function? If yes, justify your answer with a contrapositive argument. If no, justify your answer by showing how to break the property.

   i. Preimage resistance

   Solution. Yes, H is preimage-resistant assuming f is preimage-resistant. Suppose H is not preimage resistant. Then, for a random value of y, we could compute x such that $H(x) = y$. By definition of H, this would imply that $f(x' - x'') = y$ giving us a preimage $x' - x''$ for f.

ii. Second preimage resistance

Solution. No, H is not second-preimage-resistant. Suppose we are given a challenge $x = x'\|x''$ to find a second preimage of (for H). Then $-x''\| - x'$ (where the negatives are taken modulo 3) is such a second preimage:

$$H(x'\|x'') = f(x' - x'') = f(-x'' - (-x')) = H((-x'')\|(-x'))$$

Note that this solution works only when $x \neq 0$; however, since the challenge in the second-preimage-resistance experment is chosen uniformly at random from the domain, the degenerate $x = 0$ case occurs with negligible probability.

iii. Collision resistance

Solution. Since H is not second-preimage-resistant by the previous part, and every collision-resistant hash function is second-preimage resistant by a result shown in class, then H cannot be collision-resistant.

(b) [1 mark] Suppose $m = 128$, so that the input space for H is $\{0,1,2\}^{256}$ and the output space is $\{0,1,2\}^{128}$. If you sample the inputs uniformly at random, what is the expected number of steps before you find a collision in H? It is okay to use an approximation here, as long you state, and justify that approximation.

Solution. The number of possible outputs is 3^{128}. Assuming that the outputs are also uniformly distributed, the birthday paradox implies that we will need about 3^{64} tries to find a collision.

5. [7 marks] **Sponge construction**

The sponge construction is a key part in multiple cryptographic schemes, most notably the SHA-3 hash function. The construction has the following properties:

- A message m of length $n\ell$ is separated into n blocks $(m_1, \ldots, m_n)$ each of length ℓ.
- The initialization vector v is of length $\ell + r$, and is separated into a two potions $v = v_1\|v_2$, where v_1 is ℓ bits in length and v_2 is r bits in length.
- $\pi : \{0,1\}^{\ell+r} \to \{0,1\}^{\ell+r}$ is a hard-to-invert permutation.

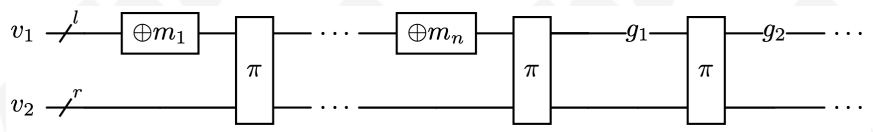
Here is the pseudocode for a sponge construction based on π that takes an input message m of length $n\ell$, and produces an output g of length $s\ell$.

$H(m)$

```
1 : h_1‖h_2 ← v_1‖v_2
2 : for i = 1, ..., n :
3 :     h_1 ← h_1 ⊕ m_i
4 :     h_1‖h_2 ← π(h_1‖h_2)
5 : for j = 1, ..., s :
6 :     g_j ← h_1
7 :     h_1‖h_2 ← π(h_1‖h_2)
8 : return g_1‖g_2‖...‖g_s
```

The first loop is called the *absorbing phase* and the second is the *squeezing phase.*

Here is a diagram showing the operation of the sponge construction:



For the sponge construction to be a secure cryptographic hash function, we need that the permutation π is hard to invert.

Note that usually the permutation π is a permutation on the entire space of all $2^{\ell+r}$ bitstrings, not a permutation on the $\ell + r$ positions of the bitstrings. This question will explore what goes wrong if that is not the case.

Suppose that π was simply the function shifting each bit one position to the right (and looping around), so that π maps $(b_1 b_2 \dots b_n)$ to $(b_n b_1 \dots b_{n-1})$). Your task is to justify why using this π would make H insecure.

(a) Prove that, if π is the shifting function as described above, then the sponge construction is not preimage resistant. For your attack, you can fix a number of input blocks $n \geq 1$ and a number of squeezing rounds $s \geq 1$ that makes it easier.

Solution. Fix $n = 1$ and let s be such that $s \geq r$. Suppose we are given an $s\ell$-bit output $g = g_1 \| g_2 \| \dots \| g_s$ to invert. Let $h_1 \| h_2$ denote $(\ell + r)$-bit state of the sponge construction immediately before g_1 is output. Since $n = 1$, we have that

$$h_1 \| h_2 = \pi((v_1 \oplus m_1) \| v_2)$$

We know that $h_1 = g_1$ by definition. We know v_1 and v_2 (since they are the fixed initialization vector), and π is invertible since it is a cyclic bit shift, so if we could learn h_2 then we would be able to compute

$$m_1 = \left(\pi^{-1}(h_1 \| h_2)\right)_{[1 \dots \ell]} \oplus v_1$$

Since π is a cyclic bit shift by 1 position, then after r applications of π, all r bits of h_2 will have appeared in the output values $g_1, \dots$. In particular, we can set the ith bit of h_2 to be the first bit of g_i, for $i = 1, \dots, r$. Now we have h_2, and as noted above we can thus find an m_1 such that $H(m_1) = g$.

(b) Prove that, if π is the shifting function as described above, then the sponge construction is not second preimage resistant. Recall that the second preimage does not necessarily need to be of the same length.

Solution. Observe that $\pi^{\ell+r}$ is the identity function. So if we are given an input m for which we need to find a second preimage, then appending $\ell + r$ blocks of 0's to m does not change the output. So $H(m) = H(m \| 0^{s(\ell+r)})$.

(c) Prove that, if π is the shifting function as described above, then the sponge construction is not collision resistant.

Solution. Since H is not second-preimage-resistant by the previous part, and every collision-resistant hash function is second-preimage resistant by a result shown in class, then H cannot be collision-resistant.

Academic integrity rules

You should make an effort to solve all the problems on your own. You are also welcome to collaborate on assignments with other students in this course. However, solutions must be written up by yourself. If you do collaborate, please acknowledge your collaborators in the write-up for each problem. *If you obtain a solution with help from a book, paper, a website, or any other source, please acknowledge your source. You are not permitted to solicit help from other online bulletin boards, chat groups, newsgroups, or solutions from previous offerings of the course.*

Due date

The assignment is due via Crowdmark by 11:59:59pm on October 10, 2024.

If you are requesting an extension or accommodation because of a verification of illness form or a self-declared student absence, see the "Missed assignment policy and forms" instructions on the CO 487/687 LEARN site.

Changelog

- Mon. Sep. 30: Fix wording about permutation π in A-2-5 parts (b) and (c).
- Thu. Oct. 3: Fix a typo in Q4.