THE UNIVERSITY of NEW ORLEANS

# CSCI 4130/5130 INTRODUCTION TO CRYPTOGRAPHY

## Fall 2024 – PA2

## *DUE: February 24, 11:59pm*

**PART 1 [30 points]**

**1-) [15 points]**

One important property which makes DES secure is that the S-boxes are nonlinear. In this problem, we verify this property by computing the output of $S_1$ for several pairs of inputs. Show that $S_1(x_1) \oplus S_1(x_2) \neq S_1(x_1 \oplus x_2)$, where "$\oplus$" denotes bitwise XOR, for:

**a.** $x_1 = 000000$, $x_2 = 000001$

**b.** $x_1 = 110111$, $x_2 = 100000$

**2-) [15 points]**

We study a real-world case in this problem. A commercial file encryption program from the early 1990s used standard DES with 56 key bits. In those days, performing an exhaustive key search was considerably harder than nowadays, and thus the key length was sufficient for some applications. Unfortunately, the implementation of the key generation was flawed, which we are going to analyze. Assume that we can test $10^6$ keys per second on a conventional PC.

The key is generated from a password consisting of 8 characters. The key is a simple concatenation of the 8 ASCII characters, yielding ($64 = 8 \cdot 8$) key bits. With the permutation PC−1 in the key schedule, the least significant bit (LSB) of each 8-bit character is ignored, yielding 56 key bits.

**a.** What is the size of the key space if all 8 characters are randomly chosen 8-bit ASCII characters? How long does an average key search take with a single PC?

**b.** How many key bits are used, if the 8 characters are randomly chosen 7-bit ASCII characters (i.e., the most significant bit is always zero)? How long does an average key search take with a single PC?

**c.** How large is the key space if, in addition to the restriction in Part 2, only letters are used as characters. Furthermore, unfortunately, all letters are converted to capital letters before generating the key in the software. How long does an average key search take with a single PC?

**PART 2 [70 points]**

**3-) [70 points]**

After her last encryption attempt ended up as a fiasco, Alice has now learnt her lesson. She vowed not to use such weak ciphers again.

Now, Alice has another need for encryption. She needs to send some confidential material to Bob. After we covered Chapter 3, she was impressed by DES and decided to use it for this task. She knew that Oscar is not rich enough to be able to brute force DES. So, she decided that a single round of DES should be good enough to keep the secret away from Bob. She did the following things:

1. Alice and Bob have already agreed upon a 64-bit secret key. (Note that this can be represented by 16 Hexadecimal characters)
2. She converted her sensitive plain text to a hexadecimal string using this encoding tool:
   https://codebeautify.org/string-hex-converter
3. She input the 16-digit hexadecimal key and her hexadecimal encoded text (which is the sensitive plain text) into this DES tool to get the encrypted text and sent it to Bob. (Use ECB mode)
   https://emvlab.org/descalc/

Oscar who was keenly spying on Alice's network got to find out that this was the cipher text that Alice sent to Bob:

39783581497F2C68D67E7FC5A62346B320239CD93DC232C1580B612054B3C5C203320B5
83D8AA7C572F4D30977CBE55DDB99FE096554DBA1

But, unfortunately for Alice, there was a small security bug in the software that Alice used. The authors of the software forgot to wipe out the variables that are used for storing the round keys during encryption. As a result, Oscar was able to log into the same computer that Alice used and obtain the last couple of DES round keys that were used from unused system memory.

These are the keys that Oscar got:

$k_{15}$ = BB9D874D36B3

$k_{16}$ = 8B1E4BA8F3A1

Given this info, will Oscar be able to find the key to decrypt Alice's cipher text? Pretend you are Oscar and try to decrypt the text. What is the text that Alice sent to Bob?

(**Hint:** The goal should be to first obtain the key that was used. For this, you can either try to code this or do this manually. Either way, you should try to retrace the steps of the key schedule, use some properties of the key schedule and obtain the necessary 56 key bits. But, it might be easier and less error-prone if you code. My Python-based solution is about 40 lines long to give you a rough estimate of the effort involved.)

**Part 2 Info:**

You need to write **a short report** and explain your solution. Your report needs to include **plain text, encryption key** as well as **any code** that you wrote as part of the cryptanalysis.

- It's also OK to do the entire cryptanalysis in a manual manner (without any programming). If that's the case, explain how you solved it in your report and show your solution step by step.
- You can also use any of the available online tools to find the solution. In that case, please **mention the tool** that you used in the report and also **how you used it** to arrive at the answer.
  - **Note:** Similar to assignment 1, you are allowed to use genAI tools (e.g chatGPT) **partially in this part**. E.g. some minor steps.
  - However, you are **not** allowed to solve entire problem via GPT. Also, do not copy paste anything.

**Final Deliverables:**

Combine your part 1 and part 2 pdfs and submit a **single pdf** file.