Cryptographic Engineering: Assignment 1

Submit a C file through Canvas

For this lab, you are going to code some functions in C and use these functions to run the same questions in Assignment 1 Prelab. A template (.c file) is created for you. Download it and complete the exercises below by filling the functions in the template. Compile the code and run it. At the end of the file, **write ALL the output as a comment**. Once completed, submit the C file through Canvas.

Remarks:

- Do **NOT** remove or add any function in the template.
- Do **NOT** modify the function name or parameters in the template.
- Do **NOT** remove, add, or modify any function in the template other than the ones stated below.
- Do **NOT** add any library to the C code.
- All variables you need have been created and initialized for you, you do NOT need any additional variable.

The main function tests the values you did in the Prelab. In addition, A test bank (contains additional tests and answers) has been created for you. This test bank will make sure that you coded the functions correctly. The only thing you have to do is copy `assign#_bank` file from Canvas into the same folder as where you run the compiled executable. It will show the values that failed. Once everything passes, it should display `bank_passed = 1`.

1. Write a function that computes the residue of any value $a < p^2$ over the following numbers. Use the methods you learned in the Prelab. You are **NOT** allowed to use % or / operators for this exercise. In the main function, $a = 2^{30} - 18 = 1073741806 =$`{0x3FFFFFEE}` is tested.
   (a) $p_1 = 2^{17} - 1 =$ `{0x1FFFF}` (Mersenne prime). To be completed in function `modp1.`
   (b) $p_2 = 2^{26} - 5 =$ `{0x3FFFFFB}` (Pseudo-mersenne prime). To be completed in function `modp2.`
   (c) $b = 2^{16} =$ `{0x10000}` (Not a prime number). To be completed in function `modb.`
2. Write a function that computes the multiplicative inverse of a over $\mathbb{F}_p$ where $p = 2^{17} - 1$ (Mersenne prime from the previous exercise) using the following methods. In the main function, $a = 51$ is tested.
   (a) Fermat's Little Theorem (FLT). To be completed in function `FLT.`
   (b) Extended Euclidean Algorithm (EEA). To be completed in function `EEA.`

   Remark 1: To receive full points for the FLT implementation, call the reduction function from exercise 1. You are not allowed to use the % operator or the / operator.

   Remark 2: Remember to perform reduction for your intermediate results as well as your final result inside your FLT implementation. Your intermediate results should also be inside the finite field (Otherwise you cannot allocate them in memory using a 64-bit data type).

   Remark 3: You are free to use any operator for the EEA implementation including the % and / operators.

3. In this exercise, you are going to get the number of cycles for each method used in Exercise 2. The method that has less number of cycles is faster. The function to compuste the number of cycles has already been created for you.

   Remark 1: The only requirement is to log the number of cycles.

   Remark 2: Make sure to run the code multiple times as sometimes your PC is running other tasks and the cycle count will not be accurate.

Grading

Exercise	Grade	Note:
1 (a)	10 pts	0 if / or % operators are used
1 (b)	10 pts	0 if / or % operators are used
1 (c)	8 pts	0 if / or % operators are used
2 (a)	10 pts	0 if / or % operators are used
2 (b)	10 pts	
3	2 pts.	To receive the full points, the actual faster function must have less clock cycles
Formatting		-5 pts. To not lose any points, readability and proper indentation are required. Follow the same formatting as the template. Indent 4 spaces when going into the body of a block. Leave an empty line when working on a new block or when you are working on a new part in your function.
Late Submission		-5 pts per day for 2 days. Afterwards a grade of 0 is given.
Total	50 pts	

Remark 1: Code that does **NOT** compile on Linux (gcc) will receive a grade of 0 for the whole assignment.

Remark 2: Students are not allowed to modify any function other than the ones stated in the exercises. Modifying the function name or parameters will receive a grade of 0 for that function. Modifying functions not stated in the exercises will recieve a grade of 0 for the whole assignment. Adding any library (such as math.h) will receive a grade of 0 for the whole assignment.

Remark 3: A few points might be deducted if the code is ambiguous. Provide comments whenever the code is not clear or too complex. A good rule of thumb to know when to comment is to ask these 2 questions: Will you be able to quickly understand and update this part of your code if you looked at it in 5 years from now or if someone else wrote the code, will you be able to understand it.