**048891 - Hardware Systems Security:  
From theory to practice**

**Assignment 1**

In this assignment, you will implement the modular exponentiation operation, explore its susceptibility to attacks and evaluate the protection measures.

Using a programming language of your choice (C, Python, Java, Matlab, Excel, etc.), implement a function that performs modular exponentiation of two numbers of a given size *s* in bits. The numbers can be represented for example as arrays of characters. It is also fine to use existing Bignum or other libraries. However, keep in mind that you will need to make modifications to the library as part of the drill. You may need to use a speed-up technique, such as the Montgomery reduction.

1. First use the basic left-to-right square-and-multiply algorithm. Measure the execution time as a function of *s*.
2. For *s* = 4096, check the dependency of the execution time on the Hamming weight of the exponent. What do you observe? What can you conclude from the observation?
3. Now let’s assign different weights to the square and the multiply operations. Each time the operation is called, its weight is appended to an array of weights. Examine the resulting trace of weights. What information can be obtained from the trace?
4. Now, add a dummy multiply operation as was shown in the class. Repeat items (2) and (3).
5. For *s* = 4096, implement the C-safe error attack that discovers the key at the end. Your code should emulate a real-life attack. Namely, the attacker can only invoke the exponentiation function and observe the output, but she has no access to the input parameters. In addition, the attacker can corrupt specific code sections.
6. Now, implement the Montgomery power ladder. Repeat items (2), (3) and (5).

**Example of the modular exponentiation function header in C**:

char \* modexp (char \* base, char \* exp, char \* N, unsigned numbits);

**Submission deadline**: Dec 15, 2024

**Submission in couples.**

**Submission contents**:

1. Report
2. Source code as a project or accompanied by build and run guidelines
3. Execution results

**Questions**:

In the class or by email: leonida AT technion.ac.il