Assignment 4

Due date: February 28, 2022

Please complete this assignment (150 pts total) and submit your report/program code (all files compressed in one `.zip` to `vincent.immler@oregonstate.edu` with subject as follows:

Subject: CS/ECE599 Assignment 4, Winter 2022, *Your Name*

1. An important processing platform for digital signatures are smartcards. Older smartcards are typically based on an 8-bit processor. Let us assume that such a processor requires one clock cycle for an 8x8 bit multiply-and-add. Please assume that a simple square-and-multiply algorithm is used and a b-bit multiplication and a b-bit squaring requires the same number of clock cycles. (30 pts)
   a) What is the minimum required processor clock speed to process an RSA signature in less than 0.75s, if the modulus N is $b = 512$ bits and $b = 1024$ bits? Please assume that the secret exponent is randomly chosen and on average contains zeros/ones equally. (25 pts)
   b) Is this a reasonable goal if the processor is not allowed to clock faster than 8 MHz? (5 pts)
2. A program needs to be developed such that fault attacks on a RSA-CRT implementation can be exploited to reveal the original RSA primes of a 1024 bit RSA key. The fault was previously generated during one CRT exponentiation and caused a faulty s' of the RSA-CRT while computing the digital signature. The wrong result s' is the important input parameter for the program. Other input parameters for the program are the public RSA parameters, moduluns N and exponent e, in addition to the correct result s and the message m to be signed. Two different use cases should be considered. (120 pts)
   a) the RSA modulus N and both the correct result s and a faulty result s' of an RSA-CRT are used (Bellcore) (60 pts)
   b) the RSA modulus N, the public exponent e, the faulty result s', and the message m are used (Lenstra) (60 pts)

   All needed input parameters are provided in two separate ASCII files. The hexadecimal representation starts with the most significant bit and ends with the least significant bit. Determine the RSA primes from the input file 'fault1.txt' and 'fault2.txt'.

   Your program should output the original RSA primes. Note: you do not have to parse the files but can copy-paste the respective values into your script.

Please write all your programs in one of the following languages/environments: Python/Jupyter. Your `.zip` file should contain your code, instructions how to make it run (if needed), the figures, etc.