HARDWARE SECURITY

ASSIGNMENT2

NAME:

Sara Muhammad Youssef

ID:

31

PART1:

- Using Montgomery modular multiplication in RSA implementation increases decryption process speed by **100%**
- In order to measure the performance speed up in the decryption process:

    1. Different size of random inputs is collected to be encrypted then decrypted in two methods:

        -RSA; using default modular multiplication.

        -Modified RSA; using Montgomery modular multiplication.

    2. Time taken by each method in each decryption process is recorded to calculate average time.

PART1 OBSERVATION:

- Using inputs of size (10, 100, 1000) sample, the corresponding average time of decryption process in millisecond for each method is:

Input size	RSA	Montgomery
10	4	2
100	32	16
1000	334	167

- Montgomery modular multiplication decreases decryption time by 50; thus, Montgomery doubles the speed of decryption process.

PART1 CHALLENGES:

- To decrease time taken by Big-Integers operations, it is important to make use of the fact that **R** “temporary modulus” is a power of 2; thus division, multiplication and modulus calculations take less time if **right and left shifting** are used instead of built-in library functions.

PART1 IMPLEMENTATION:

- Use default RSA implementation with replacing modular multiplication with Montgomery modular multiplication.
- In Montgomery; avoid repeating the same calculations on the same values, just calculate them once at the beginning
- Perform possible R “temporary modulus” calculations using shifting.

PART2:

- This table contains number of correct estimations out of **20 trials** for:
- different input size

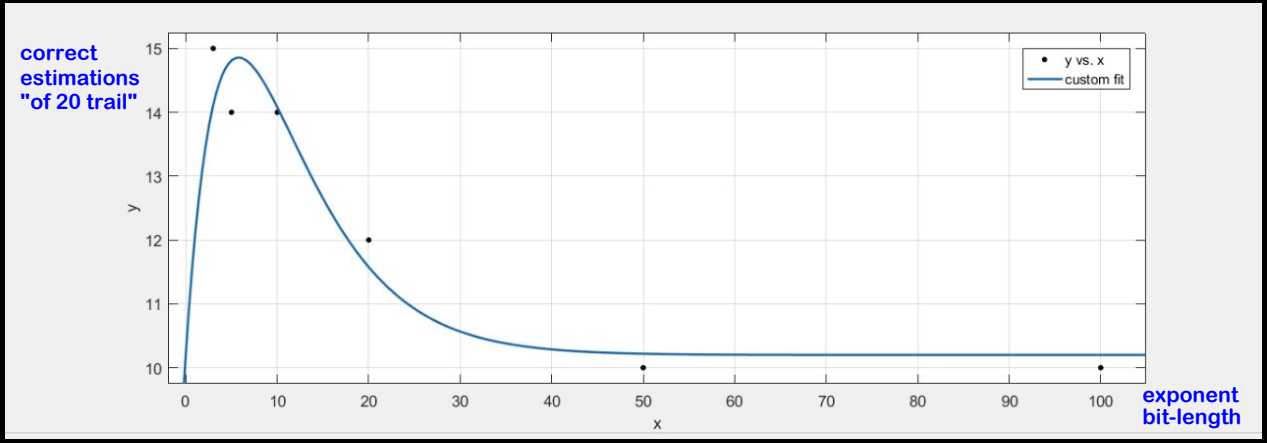
  (5000, 10000, 20000)
- different private exponent bit-length

  (3, 5, 10, 20, 50, 100)
- running every input size with each exponent 5 times and taking the average:

Exponent	3	5	10	20	50	100
5000	11	12	12	11	10	9
10000	14	14	14	12	11	10
20000	15	14	14	12	10	10

PART2 OBSERVATIONS:

- Increasing sample space above 10000 doesn’t have great effects as it almost has the same accuracy level.
- Increasing number of most significant bits in private exponent makes it harder to detect the required bit in the exponent.



PART2 IMPLEMENTATION:

- For each exponent; Generate random inputs, save them in addition to their corresponding decryption running time.
- Use 4 lists to divide measured time depending on whether current estimation is zero or one and whether reduction step is performed or not.
- Calculate the average for each list and compare their difference
- Expected bit value is assigned to the estimation that has the biggest difference between its lists average.

CONCLUSION:

- It’s more practical to use Montgomery modular multiplication in applications with repeated modular multiplication operations.
- Increasing number of exponent bits leaks fewer information in case of timing attack as modular operation takes much time that makes it harder to observe additional time taken in reduction step, hence it reduces detection accuracy.