**CSE 345/545: Foundations to Computer Security**
**ASSIGNMENT 2 (TOTAL OF 73 POINTS)**
**Deadline: Friday, September 25, 2020**

## Instructions:

- Do the Assignment individually.
- Python is the preferred programming language, submit .py files for Part II in the format <RollNo_ProblemNo>.py.
    - Example: 201556073_1_a.py
- Submit a pdf/doc file for theory/reasoning. *Do not* zip the files while submitting.
- Queries, if any can be posted on google classroom .
- Note: Answer the questions in bullet points. Keep your responses crisp and to the point.
- Please avoid plagiarism, mention any reference if taken.

## Part I

---

1. Using the RSA public key cryptosystem, if p = 13, q = 31 and d = 7, then calculate the value of e.

   **[5 marks]**
2. A cryptosystem consists of three algorithms: one for key generation, one for encryption, and one for decryption
    a. Search for the phrase “cryptanalysis” on the web and list down the requirements to break a cryptosystem?
    b. What measures/rules/parameters will you use to evaluate the strength of a cryptosystem?
    c. Look up the phrase “provable security”. What is the base assumption that all secure cryptosystems in the world holds. [Hint: Why is quantum computer feared in the security context? ]
    d. Security through obscurity (STO) is considered as the opposite of provable security. Do you think STO will survive the post-quantum world? Explain.

   **[2 X 4 = 8 marks]**
3. How is location based access control different from situation aware access control? How can you achieve both using a mobile phone?

   **[5 marks]**

## Part II

---

**Notes:**

- This part uses the Python language library GMP You are required to install the library, and run all experiments.
- Notations and algorithms in supplementary material will be the standard for this assignment.
- Deliverables for programming parts are python files in the format <RollNo_ProblemNo_Part>.py

---

1. Implement RSA in python using GMP library.
   a. Write a program to encrypt 'm' given 'p' and 'q'.
      Input: p, q, m - in each line; p<q.
      Output: c, e, d, n - each number in a separate line
      Constraints: integers p, q and m are up to 1023 digits long. Avoid using loops.
      **[10 marks]**
   b. Write a program to decrypt 'm' given 'c' and 'd' and 'n'.
      Input: c, d, n - in each line.
      Output: m
      Constraints: same as 1.a.
      Note: Values of m, c, d, n will be taken from 1.a. for evaluation.
      **[10 marks]**
   c. How did you calculate the value of 'e'? Explain why your method to solve for 'e' always returns a coprime to $\phi$
      **[5 marks]**

2. There are multiple variations of RSA in literature. One such variation is the Dependent RSA. The algorithm is given in supplementary material.
   a. Write a program using GMP for encryption:
      Input : p, q, m - in each line; p<q.
      Output: C1, C2, k, e, d, n
      Constraints: integers p, q, m can be up to 1023 digits long. Avoid using loops.
      **[10 marks]**
   b. Write a program using GMP for decryption:
      Input : C1, C2, e, d, n - in each line.
      Output: m, k
      Constraints: Same as 2.a.
      Note: Values of C1, C2, k, d, n will be taken from 2.a. for evaluation.
      **[10 marks]**
   c. Compare Dependent RSA with vanilla RSA and reason why Dependent RSA works? [theoretic proof is not expected]
      **[10 marks]**

## Supplementary material

---

1. The steps in an RSA Algorithm are
    a. Choose two prime numbers $p,\ q$.
    b. Let $n\ =\ p\ *\ q$
    c. Let $\phi =\ (p-1)(q-1)$
    d. Choose a large number $e\ \in [2,\ \phi - 1]$ that is coprime to $\phi$.
    e. Compute $d\ \in [2,\ \phi - 1]$ such that $e\ \times d\ =\ 1\ (mod\ \phi)$, and $d$ must be coprime to $\phi$
    f. $(e,\ n)$ is the public key
    g. $(d,\ n)$ is the private key
    h. **Encryption**:
        i. $C\ =\ m^e\ (mod\ n)$
    i. **Decryption**:
        i. $m\ =\ C^d\ (mod\ n)$

2. The steps in a Dependent RSA Algorithm are:
    a. The key generation process is exactly as in the original RSA.
    b. Choose a random integer $k$ in the residue class of $Z_{n^*}$
    c. **Encryption**: to obtain ciphertexts $C_1\ C_2$ given a message $M$
        i. $C_1\ =\ (k+1)^e\ mod\ n$
        ii. $C_2\ =\ m\ [k^e(mod\ n)]$
    d. **Decryption**: to get the message
        i. $k\ =\ C_1{}^d\ (mod\ n)\ -\ 1$
        ii. $m\ =\ C_2/[k^e\ (mod\ n)]$