**HW3** **CSL7480: Cryptography**

Due date: 7th March 2023. 11:59 pm.

---

**Submission process**: Submit two files (NOT as a zipped archive. Submit 2 separate files): the source codes for the second questions and a pdf file. The pdf file should contain answer to question 1, and explaination about your attack code for the 2nd question. Name your files as (i) rollno-hw3.pdf, and (ii) rollno-hw3-code (replace rollno with your entry number). Any programming language is acceptable for the problem. Mention your roll number and name clearly at the top in the pdf file.

---

1. (a) Compute the SHA256 hash of your email-id. Let it be $h_1$.
   (b) Find another IIT Jodhpur email-id (i.e. the format of the id should be x@iitj.ac.in where you need to find a suitable value for the variable x) which has the hash value $h_2$, such that $h_1$ and $h_2$ match in the first 16 bits.
   (c) Repeat the above 10 times and store the number of calls you made to the SHA256 hashing algorithm in each of these trials. If the average number of calls to SHA256 is $t$ then report the value of $\log_2 t$.
   (d) Comment on the value you reported above. Why are you getting this specific value?
   (e) Find two email-ids of the form x@iitj.ac.in and y@iitj.c.in such that their hash values match in the first 16 bits.
   (f) Repeat the above 10 times, and report the average number of trials as in part (c).
   (g) Comment on the value you reported above. Why are you getting this value? Why is it different from the one in part (c)?

2. In PKCS v1.5#1 (RFC 2313) [1], all message are padded to specific format before being encrypted. (As discussed in the class, the textbook RSA, the one without any padding, is completely insecure.) The padded message (in hexadecimal) is `0x00||BT||PS||0x00||D`, where `BT` stands for `0x02` for encryption functionality (it is `0x01` for signatures but we do not need it for this question), `PS` are randomly chosen $t$ bytes ($t$ is not a constant), with the restriction that none of the bytes are `0x00`, and `D` is the message.

   We wish to implement Bleichenbacher's attack [2] against RSA encryption in this exercise. The attack is in the CCA2 model, i.e. the attacker can query the decryption oracle with a ciphertext, and depending on the answer, decide what will be her next decryption query. Of course, the attacker can't query the target ciphertext. Since the server implementing RSA PKCS#1 would not have replied with the decrypted message, the model chosen by Bleichenbacher was to get a reply whether the decrypted message is conforming to PKCS#1 or not. That is, if the first two bytes of the decrypted message are equal to `0x00||0x02`.

   Let us say that the attacker has a taregt ciphertext $c$ and her aim is to find the corresponding message $m$. She chooses an $s \in Z_n^*$ randomly and queries the decryption oracle with $c' = c \cdot s^e \mod n$. She then verifies if the decrypted message $m'$ is PKCS conforming. If not then she attempts to find another $s$ which satisfies this property. When she succeeds, she already gets *some information* about the target message $m$. Recall that she knows $e, n, c, c'$ and $s$. She can repeat this idea till she has *complete information* about the message $m$ and she can determine it uniquely.

   You will implement the following functionality in this exercise:

(a) An RSA-PKCS#1 encryption implementation which can be given any arbitrary message $m$ to encrypt and produce the ciphertext $c$. You will call this to encrypt a message created by you, which should include your roll number. Rest of the message can be anything but at least some part of the message should be randomly chosen every time this function is called. (Your code should NOT be tailored to work on a specific message only).

(b) A decryption oracle which returns a single bit answer informing if the decrypted message is PKCS#1 conforming.

(c) An attack function which gets the ciphertext $c$ and can call the decryption oracle as many times as needed (except on the ciphertext $c$).

You have to show that you can correctly decrypt the challenge ciphertext $c$. Repeat your code a few times (for different random messages) and compute the average number of calls to the decryption oracle $q$. Report $\log_2 q$.

Note: You must generate random primes $p$ and $q$ and create your own decryption exponent for a public key $e = 65537$ for implementing part (a). You can use RSA-1048 for this exercise.

**References**:


1. RFC 2313: `https://www.rfc-editor.org/rfc/rfc2313`
2. Daniel Bleichenbacher. Chosen Ciphertext Attack against protocols based on the RSA Encryption standard PKCS#1. Crypto 1998. `https://link.springer.com/content/pdf/10.1007/BFb0055716.pdf`