556: Introduction to Cryptography
Homework - 3

1. RSA is a relatively slow algorithm, and because of this, it is less commonly used to directly encrypt user data. We can use public key system such as RSA for secure key establishment. The sender encrypts the DES encrypting key using the receiver's RSA public key and sends the encrypted DES encrypting key to the receiver. The receiver then decrypts the encrypting key using the receiver's RSA private key. Hence, DES data-encrypting keys that are encrypted using RSA public key can be exchanged safely between two people.

2. RSA critically depends on the fact that prime factorization of very large numbers takes a huge amount of time. It is based on the fact that it is easy to take two (very large) prime numbers and multiply them, while it is extremely hard to do the opposite - meaning: take a very large number, given which it has only two prime factors, and find them. The abundant number of primes helps RSA to pick private key at random. If there were small number of primes it would be easy to crack the hash function.
    a. Other requirements for RSA system:
        i. Combining RSA with some form of padding scheme to avoid practical attacks
        ii. RSA can be used to sign a message to verify the origin of a message

3. Most cryptographic hash functions are designed to take a string of any length as input and produce a fixed-length hash value and is a one-way function, that is, a function which is practically infeasible to invert. The ideal cryptographic hash function has the following main properties:
    a. deterministic: same message always results in the same hash
    b. quick to compute the hash value for any given message
    c. infeasible to generate a message that yields a given hash value
    d. infeasible to find two different messages with the same hash value

    $h(x) = x \bmod N$ where N is a 64-bit integer

    I think the above hash function is not secure enough to use in hash-and-sign signature scheme mainly because of the above requirements and the below properties:
    - Vulnerable to preimage attacks: Given a hash value h it is very easy to find message m such that h = hash(m)
    - Vulnerable to second pre-image attacks: Since we are considering N to be 64-bit integer and we are producing 64-bit long hash, I think it is easy to find second message m2 given input m1 such that hash(m1) = hash(m2).
    - Collisions of two different messages m1 and m2 may be found by birthday/rainbow attacks