CSCI 6634/4534 – Cryptography – HW 2 – 24 Spring
Due (on Canvas): 3/27/24 Wednesday 10.00 pm

- Quiz 2 <u>**in-person, open book, open notes**</u> will be on Tuesday 4/2/24 and will cover the topics from the following list which will be covered in Weeks 5,6,7,8
  - <u>MODERN SECRET KEY CRYPTOGRAPHY:</u> Double DES, Triple DES
  - <u>PUBLIC KEY CRYPTOGRAPHY:</u> public key schemes, RSA, Diffie-Hellman.
  - <u>KEY DISTRIBUTIONS:</u> key management, key distribution schemes.
  - <u>AUTHENTICATION PROTOCOLS :</u> secret key and public key protocols.
  - Please note that this quiz will cover topics covered till Kerberos, but not including Kerberos i.e. Kerberos and any topic covered after Kerberos WILL NOT be covered in this quiz

1. (15 points)
   (a) In triple DES using three keys the encryption is done as follows: $C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$. Explain precisely (in terms of K_1, K_2, K_3) how you (as the cryptanalyst) would mount a meet-in-the-middle attack on triple DES with three keys, assuming you had enough known plaintext, ciphertext pairs.
   (b) Approximately how many different keys would your attack have to try.
   (c) Approximately how much space would your attack require.

2. (15 points) A and B are going to communicate via a virtual circuit. A already knows B's public key KU_B, and B already knows A's public key KU_A. After the virtual circuit has been set up, A needs to convince B that he is A. You have to design a protocol to accomplish this task.
   - This protocol should not assume that A and B share some secret information.
   - There is no arbitrator or trusted third party i.e. all communication happens directly between A and B.
   - After the virtual circuit has been set up (which happens before the protocol starts), this protocol should have only one round for authentication purposes, and this one round will be as follows:

     Round 1: A will send a single message to B which will convince B that it is A on the other end of the virtual circuit.
   - The protocol should be designed so that it can be repeated many times i.e. A might want to communicate with B again in the future, and A, should be able to prove his identity to B using the same protocol again *without having to change keys.*
   - The protocol should be resistant to replay attacks i.e. even if the public keys, private keys remain the same, the bad guy BG should not be able to obtain any information he gains from previous iterations of the protocol to masquerade as A in the future.
   - We are only interested in authentication, not confidentiality.

   You have to
   (a) State any assumptions you are making.
   (b) Show precisely what A will transmit to B in round (1) i.e. I don't need an explanation here, just need you to show me exactly what A is sending to B.
   (c) Explain precisely in a step-by-step fashion what is the test B will run to make sure it is A on the other end (and not the BG) i.e. I don't need an explanation here, just need you to show me exactly what is the test B is running.
   (d) Explain clearly why BG cannot do a replay attack.

3. (10 points) Consider the Diffie-Hellman scheme with $q = 11$ and $\alpha = 7$.
   (a) Show that 8 is a primitive root of 11.
   (b) If A has the public key $Y_A = 6$, what is the private key X_A. Show all calculations.
   (c) If B has the public key $Y_B = 3$, what is the private key X_B. Show all calculations.
   (d) Show the calculation done by A to get the shared key K_{AB}.
   (e) Show the calculation done by B to get the shared key K_{AB}.

4. (10 points) Consider the ElGamal scheme (described in section 10.2) with $q = 11$ and $\alpha = 7$. A has a private key $X_A = 6$.
   (a) Show what will be calculations done by A to get the value of the public key Y_A.
   (b) B choses the random integer $k = 2$, and the plaintext B wants to encrypt is $M = 3$. Show what will be calculations done by B to get the value of the ciphertext C_1, C_2.
   (c) Show what will be calculations done by A to recover the plaintext M from C_1, C_2. Note that you first need to figure out what is the value of K.

5. (15 points) Give *brief and clear* explanations for the following by showing in a step-by-step manner how the BG would attack the scheme if we changed things as suggested in the question

   Consider the scheme (which we studied in class) from fig 14.12 (which is in the power point notes) for distributing public-keys.

   (a) In Step (2), if the message is sent without being signed by the authority's private key, how would the bad guy BG attack the scheme?
   (b) In Step(2), if T_1 is not included, how would the bad guy BG attack the scheme?
   (c) In Steps (3) and (6) if nonce N_1 was not being exchanged, how would the bad guy BG attack the scheme?
   (d) In Step(2), if A's original request is not included, how would the bad guy BG attack the scheme?

Programming Problem: (35 points) What you have to turn in and where you have to turn it in is similar to how you needed to turn in stuff for the HW1 programming problem. In this problem, in the HW2 submission, along with the self-critique, the easy to read source code, a printout of the output, **you also need to show how much time each of the attacks took for each of the inputs.** Please read the programming guidelines (in the course outline on Canvas) before starting to work on the programming problem. You need to read this carefully to understand what has to be turned in and how, including the self-critique.
You have to write a program (or if you find it easier, two different programs) to implement two different attacks on RSA and see which works faster. For each attack, your program will take as input the public key e, n, and a ciphertext C and produce as output the plaintext M and the total amount of time the attack took to obtain M

1. Attack 1: Here you will generate all possible values of M till you find one for which $M^e = C \bmod n$. You can use the naive and inefficient method for modular exponentiation. Here the output should consist of M.

2. Attack 2: Here you will factor n to recover the primes p, q such that $n = pq$. You will then calculate the private key d, n, and then calculate $M = C^d \bmod n$. You can use the naive and inefficient methods for factoring n, for finding d from e and $\Phi(n)$, and for modular exponentiation. Here the output should consist of p, q, d, M.

You have to run your program on three different inputs.

1. $e = 7, n = 15, C = 10$. It is possible that this problem gets solved so fast that you do not get very meaningful timing results for this example.

2. $e = 13, n = 527, C = 356$. Again, it is possible that this problem gets solved so fast that you do not get very meaningful timing results for this example.

3. An input generated by you - you should generate the largest values for which you can get results in a "reasonable" amount of time. Here you have to start from scratch in terms of finding p,q,e,d etc i.e. you have to find these values.

Notes:

1. As discussed in class, you will need to do the mod operation frequently to keep the numbers small so as to avoid overflow errors.

2. You will have to figure out some kind of clock function to do the timing calculation.

Extra Credit Problems: 9.5, 9.7, 9.15, 10.4, 11.1, 11.4. 14.6

Extra Credit Problem 3: Implement a brute force attack on Diffie-Hellman and show how it works on some examples.