Assignment 5 – Due Thursday, April 7

1. ElGamal encryption was described in Section 10.2 of the textbook. In this problem, you are to explore how ElGamal measures up in terms of the formal security models we discussed.

   (a) It is impossible for "textbook RSA" (meaning RSA without any padding or randomness) to be IND-CPA secure. Why is that, and does ElGamal have the same problem?

   (b) Let (C_1, C_2) be an ElGamal ciphertext, computed as shown in Figure 10.3 (page 293). Write out formulas (in terms of α, k, Y_A and M) for $\alpha \cdot C_1$ and $Y_A \cdot C_2$, where operations are preformed mod q.

   (c) What result would be produced if you ran fake ElGamal ciphertext $(\alpha \cdot C_1, Y_A \cdot C_2)$ through the ElGamal decryption function?

   (d) Use these observations to create an attack algorithm that wins the CCA game against ElGamal. In addition to describing the algorithms, remember to analyze the advantage of your adversary in this game.

   As an aside, the message you should learn from this problem is that textbook RSA cannot be IND-CPA secure, much less IND-CCA secure. ElGamal in fact does turn out to be IND-CPA secure, but it is not IND-CCA secure.

2. Consider two 1024-bit large primes p and q such that $n = pq$ is a 2048-bit modulus that we want to use for RSA. Since the exponent e in RSA is a public value (it is part of the public key) it is common to use small, specially chosen values of e so that encryption is particularly fast.

   (a) In the "exponentiation by repeated squaring" algorithm, if e is chosen randomly (a random 2048-bit exponent), what is the average number of modular multiplications that would need to be performed to do an RSA encryption? (*Hint: If you look at the recursive version of this algorithm, at each level of recursion either 1 or 2 modular multiplications must be performed, depending on whether the exponent at that level of recursion is even or odd. The probability that the exponent is even is exactly the probability that a specific bit of the original, random exponent is 0, which should should be* $\frac{1}{2}$.)

   (b) If we always use $e = 3$, how many modular multiplications are needed for an RSA encryption? How much faster is this than the general case from part (a)?

   (c) What restrictions does the modulus n (or the primes p and q) need to satisfy in order to use $e = 3$?

   (d) If we always use $e = 65537$, how many modular multiplications are needed for an RSA encryption? How much faster is this than the general case from part (a)?

   (e) What restrictions does the modulus n (or the primes p and q) need to satisfy in order to use $e = 65537$?

(f) (Extra Credit!) Alice would like to send the same message M to three different recipients, all of whom have their own RSA public keys with 2048-bit moduli and a fixed exponent $e = 3$. In other words, she sends out messages $c_1 = M^3 \bmod n_1$, $c_2 = M^3 \bmod n_2$, and $c_3 = M^3 \bmod n_3$. Eve captures these messages, and can decode the message M even without knowing the private keys. How? (*Hint: The Chinese Remainder Theorem from Section 8.4 is very useful!*) If the messages are padded independently with OAEP before encryption with RSA, does this fix this flaw? Explain.

3. The "man-in-the-middle attack" for the Diffie-Hellman Key Exchange (described starting on page 290) is a serious problem that stems from the fact that Alice cannot guarantee that the message she sends to Bob is the same as the message that Bob receives. Consider using a public web site (like a discussion forum) as follows: Alice and Bob post their values Y_A and Y_B on this site, and they retrieve the other side's value from this site. Alice and Bob can also check the web site (from different hosts if they want to disguise their identity) to see if the correct value is being provided by the public web site. Does this solve the problem? Address issues such as how you identify another user, whether the communication link that Alice and/or Bob uses is compromised, etc.

4. Imagine that a very powerful *passive adversary* is capturing all Internet traffic and archiving it (perhaps in a big data center in Utah) in case it is useful at some point in the future. At some point a computer is compromised, and all data from that computer is retrieved, including secret keys. If communications that were archived from *prior to this compromise* remain secure, and can't be decrypted even with knowledge of the stolen secret keys, we say the communication protocol that was used has *forward secrecy*. Consider the following two methods for an encrypted session:

   Using RSA. At the beginning of the year a server generates an RSA keypair and sends the public key to all clients (this is a reliable key distribution, and can't be corrupted). To establish an encrypted session later in the year, the client generates a random AES session key, encrypts it with the server's public key, and sends the ciphertext to the server. The server then uses its private key to decrypt the session key, and then the client and server send all subsequent communication encrypted by AES with the session key.

   Using Diffie-Hellman. In this scenario, there are no pre-distributed keys, and the client and server perform a Diffie-Hellman key exchange as shown in Figure 10.1 in the textbook. After the key exchange, both client and server take the least significant 256 bits of the generated shared secret and uses it as an AES key for the rest of the session.

   For each of these two methods, answer the following questions (with justification – if it is insecure, your justification should include a clear description of how to break the security).

   (a) Does the method have forward secrecy?

   (b) Is the method secure against man-in-the-middle (active) attacks?

5. Consider the elliptic curve $E_{13}(1, 1)$; that is, the curve is defined by $y^2 = x^3 + x + 1$ with a modulus of $p = 13$.

   (a) Determine all of the points in $E_{13}(1, 1)$. (*Hint: Start by calculating the right-hand side of the equation for all values of x.*)

   (b) Compute the sum $(4, 2) + (8, 12)$ in $E_{13}(1, 1)$.