The University of North Carolina at Greensboro — Handout 5
CSC 580: Cryptography and Security in Computing — March 19, 2013
Prof. Stephen R. Tate

Assignment 3 – Due Tuesday, April 2

1. A key recovery attack takes a matching plaintext/ciphertext pair (P, C) and finds a key K such that $E_K(P) = C$. We can formalize this by saying that there is a probabilistic polynomial time algorithm $\mathsf{findkey}(P, C)$ that computes K. Show that if such a function exists, then the encryption scheme is not IND-CPA secure (to do this, define the adversary functions $A_1^{\mathcal{E}}$ and $A_2^{\mathcal{E}}$ for the chosen-plaintext game, where the adversary functions can call both the encryption oracle and the $\mathsf{findkey}$ function). In addition to giving the algorithms, analyze the advantage of your adversary algorithm under the assumption that $\mathsf{findkey}$ always succeeds.

2. In the handout on security models, it was shown that no stateless, deterministic encryption can be IND-CPA secure. For this problem, consider CBC mode in which the IV is picked in a way that can be predicted by the adversary (but may be stateful and/or non-deterministic).

   (a) Describe CBC mode using algorithm specifications (both encryption and decryption), where the IV is selected during encryption using a function named $\mathsf{getIV}(\lambda)$. You need to give names to both full CBC as well as individual block cipher encryption/decryption functions.

   (b) Assume the adversary has a $\mathsf{predictIV}()$ function that always predicts the next IV that will be returned by $\mathsf{getIV}(\lambda)$ (and hence the next IV that will be used in an encryption). Define adversary algorithms $A_1^{\mathcal{E}}$ and $A_2^{\mathcal{E}}$ that can win the CPA game. (Hint: Think about what is fed into the block cipher as plaintext. Can you arrange it so that you can do multiple encryptions in which the same values are fed into the block cipher?) Do a quick analysis of your algorithms to give the advantage of your adversary.

   (c) What if the $\mathsf{predictIV}$ algorithm isn't perfect? In other words, what if $\mathsf{predictIV}$ only predicts the correct IV with probability p? What is the advantage now?

   (d) CBC-Chain mode is a variant of CBC mode in which the encryption function keeps a record of the last ciphertext block produced, and uses that as the IV for the next encryption (so a series of CBC-Chain encryptions is the same as a single long CBC encryption). Show that CBC-Chain mode is not IND-CPA secure. (Note: This attack appears in the real world as part of the basis for the BEAST attack on SSL, discovered in 2011.)

3. It's clear that repeating a deterministic block cipher, as in ECB mode, is not secure. But what about repeating a secure (non-deterministic) cipher? In other words, let $E_K(P) \rightarrow C$ be an IND-CCA encryption scheme, and then define a two-block version of this by $E2_K(P_1, P_2) \rightarrow (C_1, C_2) = (E_K(P_1), E_K(P_2))$. It turns out that this construction is IND-CPA secure, but *not* IND-CCA secure. Prove the second part of that statement (in other words, give an adversary that wins the CCA game against the $E2$ two-block encryption scheme — like almost all CCA attacks, the trick is to disguise your decryption oracle requests so they don't exactly repeat the challenge ciphertext). Make sure you analyze the advantage of your adversary!

4. (*Note: Taking large modular powers is tricky, but modern programming languages have good support for this — for example, in Java you can use the* `modPow` *function in the* `BigInteger` *class, and in Python you can use the built-in* `pow` *function, where* `pow(a,x,n)` *computes* $a^x \bmod n$.) Consider the value $n = 8911$ (this is a composite number with factors 7, 19, and 67).
   (a) Select three different random a values in the range $2, \ldots, 8909$ that are relatively prime to n, and calculate $a^{n-1} \bmod n$ for each of these three a values. Does it seem that n behaves like a prime number as far as Fermat's Little Theorem is concerned?
   (b) Use your random a values from part (a) to run the Miller-Rabin primality-testing algorithm on n, showing each step. If your first a value returns "composite" you can stop with just that one simulation — otherwise, try the other a values until you get "composite."
5. Use Table 8.1 on page 246 to pick two random primes (p and q) in the range $1500, \ldots, 2000$. Compute $n = pq$ and $\phi(n)$. Pick a random a in the range $2, \ldots, n-1$ such that it is relatively prime to n. Compute $b = a^{\phi(n)-1} \bmod n$. Finally, compute the product $a \cdot b \bmod n$. What does this tell you about the relation between a and b? Will this relationship always hold for any values that you pick according to these directions? Justify your answer.
6. What are the primitive roots of 31? (A very simple program can quickly solve this problem.)
7. Is there any need for an "encryption oracle" when talking about chosen plaintext attacks on a public-key encryption system? Explain why or why not.
8. For the values you picked in problem 5, let's use b to represent $\phi(n)$ (i.e., $b = \phi(n)$). Find the prime factorization of b and then use the equation from Problem 8.12 (page 264) to find $\phi(b)$. Pick a random e that is relatively prime to b, and use the technique from problem 5 to compute the multiplicative inverse of e modulo b — call this d. Finally, verify that these values work as RSA keys: Pick two different random messages m_1 and m_2 in the range $0, \ldots, n-1$ — for each one, compute $c_i = m_i^e \bmod n$ and $r_i = c_i^d \bmod n$, and if everything works you should have $m_i = r_i$.
9. OAEP, illustrated in Figure 9.10, is a very important technique — without it, the RSA encryption function is deterministic and stateless, and so cannot be IND-CPA secure! However, using OAEP (or any technique that adds non-determinism) reduces the size of plaintexts that can be encrypted. Assuming that OAEP is using a 160-bit seed and a hash function that produces a 160-bit value, how large a plaintext can be handled using RSA with 2048-bit key? Note that you can't really get enough information from the brief description in the book to determine this — search and find a more precise description of OAEP online (and make sure you cite any source that you use!).
10. The "man-in-the-middle attack" for the Diffie-Hellman Key Exchange (described on page 305) 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.