The University of North Carolina at Greensboro — Handout 7
CSC 580: Cryptography and Security in Computing — April 18, 2013
Prof. Stephen R. Tate

Assignment 5 – Due Tuesday, April 30

Notes: This assignment is optional — you may use it to replace your lowest assignment grade if that would benefit you. Whether you do the assignment for a grade or not, these are good problems to test your understanding when preparing for the final exam. Also note that the due date is not a class day, so you will need to drop your assignment off at my office before noon on April 30.

1. The SSH protocol can use digital signatures to perform authentication without having to enter a password. Let's say you want to be able to log in from your home system to two remote systems, RemoteA and RemoteB. You create an identity on your home system, which is a keypair: the signing (private) key is kept on the home system only, while the verification (public) key is copied to both RemoteA and RemoteB. When you connect to RemoteA to log in, it sends a random challenge nonce to your home system, which is signed and sent back to RemoteA. Since RemoteA has the verification key, it can verify the signature to authenticate you.

   (a) If an attacker gains full access to system RemoteB, what information is stored on that system that is potentially compromised?

   (b) With this information, can the attacker log in to RemoteA? Why or why not? Be as precise as possible, referring to the security properties of digital signatures (page 399).

   (c) With this information, can the attacker log in to your home system? Why or why not?

   (d) What if the access to RemoteB included the ability for the attacker to monitor the plaintext of all communication with RemoteB — does that change the answer to any of the preceding questions?

   (e) Now consider a scenario in which basic passwords are used for authentication. As before, the same authentication information (in this case a password) is shared between the two remote systems. If the attacker has fully compromised RemoteB in this scenario, is access to any other system possible?

2. Consider the following authentication scheme, which is a slightly simplified version of standard HTML digest authentication. Using a secure cryptographic hash function H (meeting all security goals given in Table 11.1), if the user with user id u has password p, then the server stores $y = H(u\|p)$ in its user database. Then when a user seeks to authenticate, the server generates a random nonce n which it sends to the client system, which in turn gets a user name u and password p from the user, computes $z = H(H(u\|p)\|n)$ locally and sends the result back to the server. The server can then compute $H(y\|n)$ and see if these values match. Answer the following questions, with explanations (they're not really just yes/no questions!).

   (a) An eavesdropper sees both n and z as they are transmitted. Can the eavesdropper figure out the user's password?

(b) An attacker breaks into the server, copies the authentication database, so learns the value y. Can she compute the user's password from this?

(c) If an attacker obtains the server's user database, and consequently learns y, can the attacker log in to the system as user u?

3. The previous question gave an example of a challenge-response authentication protocol. Fred announces that he has a new authentication protocol where the server doesn't need to remember the nonce n while it is waiting for a response. In Fred's protocol, the server again generates a nonce n but then MACs n using a secret key K that only it knows, as $c = MAC(K, n)$. Fred calls c an "authenticator", since it authenticates that that n was indeed generated by the server. Both n and c are sent to the client, which operates as in the previous question, except that when z is sent to the server, the client also sends back n and c. Upon receiving these values, the server checks the authenticator c to make sure that n was a value it generated (it didn't store it!), and then proceeds as before. Unfortunately, there's a problem with this authentication protocol — what is it? What kinds of solutions can you think of?

4. Joe deals with very sensitive information, and he is worried that a virus might get on his system and compromise everything, giving an attacker access to all his files and even the things he types such as passwords. He needs to be able to sign electronic files, but if the signing key is kept on his system the virus could access the key and sign anything the attacker wanted. Even if the user had to type a passphrase to unlock (i.e., decrypt) the signing key, the virus could monitor those keystrokes and subsequently unlock the private key for its own nefarious purposes. Joe's solution is to use a smartcard: a device that can be plugged in to his system that does the actual signing. The data is sent from the system to the smartcard, the card signs it, and the signed value is returned to the user for transmission. The signing key never leaves the smartcard, which is locked down tightly and not vulnerable to viruses. While this makes matters harder for the virus/attacker, the attacker can still get signatures on whatever data it wants. How? Can you think of a solution to this problem? (Warning: just speculate a little on a possible approach — there is no really good solution known for this problem, so don't get stuck for too long trying to design a great solution!)

5. Clearly a hash function that has the strong collision resistance property also has weak collision resistance. What about the next step down? Does a hash function that has weak collision resistance also satisfy the one-way property (see Table 11.1 on page 336 for these terms)? To answer this question, consider a hash function $H(x)$ that produces k-bit hash codes, and satisfies all three of these security properties. Now construct a hash function $H'(x)$ that produces $(k+1)$-bit hash codes as follows: If x is exactly k bits long, then output $0\|x$ (a single 0 bit followed by x); otherwise output $1\|H(x)$ (a single 1 bit followed by the H-hash code of x). Is $H'(x)$ weakly collision resistant? Is it one-way? Justify your answers!

6. 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" 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 307). 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.

7. Alice uses a stream cipher that generates a "key stream" of bytes $k_1, k_2, \ldots, k_n$ that is exclusive-ORed with the plaintext $p_1, p_2, \ldots, p_n$ to produce ciphertext bytes $c_1, c_2, \ldots, c_n$. Imagine that Alice shows you the first part of the plaintext (say it's a header), and you also have access to the ciphertext. You make a suggestion regarding the header that will be followed by the unsuspecting Alice and will allow you to decode the entire message *regardless* of how secure the key stream is! Here's how: you tell Alice to delete the first character & (or something else in the header) because it's not needed, and then re-encrypt the text. Alice obliges by decrypting her file, deleting the & from the header, and then re-encrypts with the same key stream. If the deleted character is originally in position m (which you know, since you've seen the header), then she is encrypting the plaintext $p_1, p_2, \ldots, p_{m-1}, p_{m+1}, \ldots, p_n$. Show how you can use the two ciphertexts, along with knowledge of the position m and character &, in order to decode the entire message!

8. This is a programming problem. The standard way to compute multiplicative inverses modulo p is to use the Extended Euclidean Algorithm, as described in Chapter 4. When the modulus p is prime, you can also compute $a^{p-2} \bmod p$, which will give the multiplicative inverse of a.

(a) Why is $a^{p-2} \bmod p$ the multiplicative inverse of a? The reason is pretty simple, so don't give a long convoluted argument — a single, unambiguous sentence is all you need!

(b) Implement both of these methods for large (1000+ bit) integers. Note that if you are careful about your choice of programming language, you can use a built-in function for the modular powering so that this second implementation would be a single line!

Once you are sure your implementations work correctly, time them on a 1024-bit modulus. To do this, you'll need a 1024-bit prime — the best way for you to do this would be to find a way to generate your own large random prime, either by typing up the Miller-Rabin primality testing algorithm, or using any of the mathematics packages that have this functionality built in. If you really can't generate your own large random prime, you can use the one linked from this assignment on the class web page (you'll still have to get it into your program!). Note that your programs should be too fast to accurately time directly without using some advanced profiling tools. The easiest way around this is to iterate your test as many times as necessary so that the overall time is around 30 seconds. If you iterate n times, just divide your final time by n and you've got the time for one modular inverse calculation.