CS 5490/6490: Network Security – Fall 2015
Homework 1
Due by 1:25 PM MDT on Monday September 14th 2015

- **cs5490 = 17 points, cs6490 = 27 points**
- **Programming Questions (for both cs5490 and cs6490) = 20 points**
- **No cheating will be tolerated. No copying from the Internet is allowed.**
- **No extensions will be granted.**
- **The code for programming questions and the output files must be submitted using *handin* (http://www.cade.utah.edu/faqs/what-is-handin-and-how-do-i-use-it-to-turn-in-my-assignment/). The rest of the homework must be submitted as hard copies (bring those to the class or drop them off in the drop box outside of the SoC main office).**

Question 1 (*5 points*):

(a) (General) (*1 point*): How can cryptography users benefit more in comparison to the adversaries from increasing processing power?

(b) (Block Cipher) (*1 point*): In secret key-based block ciphers, do you expect to use more rounds when the block size is large? Explain briefly. No credit without explanation.

(c) (One-time Pad) (*2 points*): A one-time pad must never repeat. Yet, in a series of fixed-length numbers, for example 8-bit bytes in RC4, some number must ultimately repeat, because there are only 256 8-bit numbers (0-255). Why are these two statements not contradictory?

(d) (One-time Pad) (*1 point*): When a stream cipher like RC4 is used, show how an attacker can change the message m if the attacker knows m and the corresponding ciphertext c?

Question 2 (DES) (*2 points*): The mangler function in DES takes a 32-bit data and a key as inputs to produce a 32-bit output. Is it possible to replace the DES mangler function with a secure message digest function (e.g., SHA256)? Will there be a problem if the secure message digest produces a longer than 32-bit output (given that DES is expected to produce a 32-bit output)? Explain your answers for full credit.

Question 3 (Modes of Operation) (*5 points, each part – 1 point*): Consider a modified cipher block chaining (MCBC) scheme where $m_{n+1} \oplus c_n \oplus m_n$ is encrypted to yield c_{n+1} (recall that in the regular CBC, $m_{n+1} \oplus c_n$ is encrypted to yield c_{n+1}).

(a) With regular CBC, if one ciphertext block is lost, how many plaintext blocks are lost?

(b) With MCBC, why do things get back in sync if c_n and c_{n+1} are switched?

(c) How about if a ciphertext block is lost?

(d) How about if ciphertext block n is switched with ciphertext block n+2?

(e) How about any permutation of the first n blocks?

Question 4 (Hash Functions) ***5 points*****:**

(a) (*2 points*) Find the approximate number of messages (n) that need to be tried before finding two that had the same message digest (size k) with probability 0.6. You need to find n as a function of k. What is n when $k = 2^{256}$?

(b) (*3 points*) Problem 3, page 144, chapter 5.

Question 5 (Required for cs6490 students, extra-credit for cs5490 students) ***10 points (each part – 2.5 points)*****:** Read the following paper - Jacobson, V.; Smetters, D. K.; Thornton, J. D.; Plass, M. F.; Briggs, N.; Braynard, R. "Networking named content," Proceedings of the 5th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT), December, 2009. (You can obtain this article for free if you go through any of the University of Utah computers or use the library online resources after login to your CIS account.)

1. Contrast IP forwarding model with Content Centric Networking (CCN) model.
2. How does CCN prevent a distributed attack where an attacker requests the same content name from many different networks?
3. How does a CCN publisher control where its content travels?
4. Could CCN prevent spamming (e.g., email spamming)? Explain briefly.

Programming Questions ***20 points*****:** Using the *handin* utility (cs5490 students should also use cs6490), electronically turn in your code along with the output files for the following two parts. While you can use any computer to work on your programs, make sure that these run on the CADE lab machines. We cannot grade any programs that do not run on the CADE lab machines. You can write your code in one of the following languages: C, C++, Java, or Python.

(a) (*5 points*) Implement the RC4 code given on page 93. Try out the code with different key values to convince yourself that the code does generate random octets. [This code has a minor bug that you must fix otherwise you will see a lot of zeros.] Using the key "bcdef", and ignoring the first 512 octets, encrypt the message "I think I might enjoy this class after all." and turn in your code as well as the encrypted output.

(b) (*15 points*) Program a secret key encryption and decryption method based on the following:

- Your program must take an array of 8 characters as an input and output an array of 8 encrypted characters.
- Write the code necessary to create 8 unique random substitution tables, one for each character of the array.
- Your methods must use a 64-bit key (can be represented as another array of 8 characters) derived from an 8-character password.
- Encryption algorithm: Take the input array and *xor* it with the key. Using the *xor*ed output, perform a character-by-character substitution using the different substitution tables. Perform the permutation step once after the substitution step. For the permutation step, (circular) shift the bit pattern by one to the left with the leftmost bit becoming the rightmost bit. Repeat the above steps 16 times,

corresponding to 16 rounds, with the output of a round serving as the input for the next round.

- Decryption algorithm: Reverse the encryption algorithm. The permutation, however, should (circular) shift the bit pattern by 1 bit to the right. The substitution tables are used for reversing the substitution.

Take a sample input bit pattern (represented as an array of 8 characters) and produce the encrypted output pattern. Feed the encrypted output pattern to your decryption function to obtain the input bit pattern. Match the two patterns to ensure that your decryption indeed reverses your encryption. Do the same by changing only one bit in the input pattern. You must include print statements in your code to print the input, per round encrypted output, and per round decrypted output in an output file for both input bit patterns. The output file must also be submitted with the code.