Name:

Computer Security I: Homework 3 - Comments

Question 2 amended September 13.

Due September 17, 2010 on compass:

25 points – 6 points each section

1. Consider using AES-192 in cipher feedback mode to encrypt plaintext m:

a. What are the data must be communicated to the peer to ensure he can decrypt the ciphertext?

The encrypting algorithm, the ciphertext, the key, the initialization vector.

I'm taking off a point for IV and algorithm. The other elements I'm taking off two.

b. What information must be kept private?

The key. The other elements do not need to be kept private. Deduct 2 points for each additional item listed. They get no points if key is not listed.

c. If an error is introduced in the plaintext, how much information is lost?

The plaintext in error will be lost. Since the flawed plaintext will be fed into the algorithm, the decrypted cipher text will also be flawed. The same flawed ciphertext will be feed into the keystream generation portion of CFB. Therefore, the same key stream will be generated on encrypt and decrypt. So only the plaintext that was in error to start will be in error at the end.

d. If an error is introduced in the ciphertext, how much information is lost?

The key stream should resync itself after the faulty ciphertext bit has worked its way through the register. This should be 128 steps (the register is the same size as the algorithm block size). So 128 bits would be lost. Or 129 if you count the faulty ciphertext bit itself.

A lot of folks are saying 2 blocks. That's the answer for CBC. I'm taking away 4 points for that.

Name:

25 points. 10 points for the maximum period. 15 points for the 12 stages.

2. Let the function *f* be a six-stage LFSR with a tap sequence T=110011 and the initial value of the register is R=100100.

   Amended: Draw out 12 stages of keys and register values. Identify the maximum number of key bits that could be generated in a 6 stage LFSR before the register values start repeating. If you do find the actual period/cycle size for this linear system, you will earn an extra 5 points.

Maximum period is 2^6 – 1. Take away two points if they are off by one. Turns out this system had the maximum period.

Register	Ki	Computation	New bit
100100	0	11 xor 01 xor 00 xor 10 xor 01 xor 01	1
110010	0	11 xor 11 xor 00 xor 00 xor 11 xor 01	1
111001	1	11 xor 11 xor 10 xor 00 xor 01 xor 11	1
111100	0	11 xor 11 xor 10 xor 10 xor 01 xor 01	0
\011110	0	01 xor 11 xor 10 xor 10 xor 11 xor 01	0
\001111	1	01 xor 01 xor 10 xor 10 xor 11 xor 11	0
\000111	1	01 xor 01 xor 00 xor 10 xor 11 xor 11	0
\000011	1	01 xor 01 xor 00 xor 00 xor 11 xor 11	0
\000001	1	01 xor 01 xor 00 xor 00 xor 01 xor 11	1
100000	0	11 xor 01 xor 00 xor 00 xor 01 xor 01	1
110000	0	11 xor 11 xor 00 xor 00 xor 01 xor 01	0
\011000	0	01 xor 11 xor 10 xor 00 xor 01 xor 01	1

Taking away 3 points for folks who have the right idea but make a silly error on the table.

3. This question involves working with AES and DES encryption. Use the "openssl" utility that comes with the OpenSSL distribution to perform the following operations. OpenSSL is installed on the csil-linux machines which all members of the class should be able to access. remlnx.ews.uiuc.edu is configured for remote terminal access. "openssl" should be on your path. "openssl help" and then "openssl <keyword> help" will recursively give you more information about valid options.

   OpenSSL also runs on Windows platforms. I can provide a built OpenSSL tree from a Windows XP platform (should probably work on Vista too) on request.

Name:

http://www.openssl.org/docs/apps/openssl.html is the root man page for theS openssl tool.

50 points total. 16 points each section.

a. Fetch an encrypted file and a key file from http://www.cs.illinois.edu/class/fa10/cs461/assignments/hw3-enc-fa10.bin and http://www.cs.illinois.edu/class/fa10/cs461/assignments/hw3-key-fa10.hex . Decrypt the file using 128 bit AES in ECB mode. Look at the "openssl enc" command. It should result in a plain English file. Submit the resulting plaintext. Note: the current version of openssl seems to insist on an IV file for all modes of AES. Just use all 0's for the initialization vector.

This is the balcony speech from Romeo and Juliet. Pretty much an all or nothing section.

b. Select an ASCII text file to encrypt using AES in CBC mode. Use a non-zero IV. Submit the encrypted file, key file, size of key, IV, and any additional information we will need to decrypt the file. Do not include your binary file in the word document. Rather attach at least the binary file (i.e. the encrypted file) as a separate file in compass.

They should provide, cipher text, key, and iv. Hopefully the key and IV are in hex. If they provide them in another form we can use, that's ok. They just need to explain the form. Take away 2 points for each missing element. Obviously if ciphertext or key are missing they will fail the decrypt.
If they don't provide IV, you should still be able to decrypt with IV=0, but the first few bytes will be messed up. 10 points if you can successfully decrypt.
Some people provided a key phrase (password) instead of a key. This lost 3 points.

c. Use the "speed" option for the "openssl" command to compare the performance of AES in CBC mode using key lengths 128, 192, and 256 bits, DES in CBC mode, and triple DES (DES in EDE mode). The "speed options reports how many bytes are processed in a fixed amount of type. Report your results in bytes per second as reported by the "openssl speed" command. Rank your results from fastest to slowest.

They should provide a nice table showing algorithm and bytes per second or time. As long as all 5 algorithms are reported, they should get full points. If they don't rank in order from fastest to slowest but have everything else, take off 2 points.
AES-128 was the fastest. DES-EDE was the slowest.

Name:

Extra information on AES/DES implementations and invocations

The information in this section is not required to successfully complete your homework. Rather it is here if you wish to work with AES or DES a bit more.

Looking inside AES

If you are interested in looking at the inside of a C++ AES implementation, look at the references library at http://www.cs.uiuc.edu/class/fa07/cs461/aes-files.zip. This library includes a test AES program. It also includes a makeKey program which creates a key of the specified length using the rand() pseudo-random function.

Using Encryption in your programs

In question 3 we are using the crypto library of the OpenSSL implementation through the standard openSSL utility. You can write programs that directly invoke the crypto library API's to build your own encrypting applications. An example client program that uses AES 256 to encrypt and decrypt a string is posted at http://www.cs.uiuc.edu/class/fa08/cs461/example_crypt.cpp

On the ews machines the program is compiled by "g++ -o test example_crypt.cpp -l crypto" and invoked with no arguments. The program is currently hard-coded to use aes_256 in CBC mode. There are some man pages that define the OpenSSL crypto messages.

.