Name:

Computer Security I: Homework 3 – Answers and Comments

Due September 18, 2009 on compass:

1. *(33 points, 8 points per section. One point for free)* Consider using AES-192 in cipher block chaining mode to encrypt plaintext m:

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

(+2 points per good item identified)
Peer must have access to ciphertext, key, encryption algorithm (and related information like key and block size), initialization vector.

b. What information must be kept private?

Key must be private. All of the other information can be made public

A number of folks also noted that the plaintext should be kept private.

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

There was confusion on my part for this question. The plaintext error will propagate and alter all future ciphertext blocks. In lecture, I pointed this out and declared that the plaintext error would never sync up and it would propagate through the entire message.

But this isn't true from a practical perspective. If the "errored" version of the cipher text is sent to the recipient, the new version of the ciphertext will be xor'ed out of the stream after the block is decrypted. Thus the original version of the plaintext would be recovered.

Therefore, the right answer is that only the errored plaintext bits will be errors for the recipient. However, we gave full points for either answer.

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

*At most two blocks of information will be lost. (128*2 bits)*

Name:

2. (33 points) Let the function *f* be a four-stage NLFSR be $f(r_0...r_{n-1}) = (r_0$ and $r_1)$ or r_3 and the initial value of the register is 1001. Derive the initial sequence and cycle size.

Initial sequence:
R0: 1001 K0: 1
R1: 1100 K1: 0
R2: 1110 K2: 0
R3: 1111 K3: 1
From this point on we are stuck at 1111 with key 1

The period or cycle size is 1.

A few people miscomputed the function (e.g., used bit 2 instead of bit 3).

Other people used the non-repeating initial sequence as part of the cycle and got the period wrong.

3. (34 points) 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. csil-linux-ts1.cs.uiuc.edu and csil-linux-ts2.cs.uiuc.edu are 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.

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

   a. Fetch an encrypted file and a key file from http://www.cs.uiuc.edu/class/fa09/cs461/assignments/hw3-enc.bin and http://www.cs.uiuc.edu/class/fa09/cs461/assignments/hw3-key128.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.

   Plain text should be as below. Almost everyone who attempts this question got full points for part a.

Name:

"Fortune is arranging matters for us better than we could have shaped our desires ourselves, for look there, friend Sancho Panza, where thirty or more monstrous giants present themselves, all of whom I mean to engage in battle and slay, and with whose spoils we shall begin to make our fortunes; for this is righteous warfare, and it is God's good service to sweep so evil a breed from off the face of the earth."

"What giants?" said Sancho Panza.

"Those thou seest there," answered his master, "with the long arms, and some have them nearly two leagues long."

"Look, your worship," said Sancho; "what we see there are not giants but windmills, and what seem to be their arms are the sails that turned by the wind make the millstone go."

"It is easy to see," replied Don Quixote, "that thou art not used to this business of adventures; those are giants; and if thou art afraid, away with thee out of this and betake thyself to prayer while I engage them in fierce and unequal combat."

b. Select an ascii text file to encrypt using AES in CBC mode. 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.

Most people got this section too.
Some people omitted identifying the iv, which generally meant that they assumed an iv of 0. Others didn't provide enough information to let us figure out how to decrypt.

Should be able to use a command such as
openssl enc -in enc-file.bin -d -K sdfsdfasfsadfsadf -iv sfdsdfsdasdfsdfsfd -aes-128-cbc
To decrypt the file.

c. (12 points) Use the "speed" option for the "openssl" command to compare the performance of AES in ECB mode using key lengths 128, 192, and 256 bits, DES in ECB 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.

Name:

(2 points per algorithm plus two points for free)

I told the newsgroup to use CBC mode since ECB mode didn't seem to be available in the version of openssl we're using this year.

Here's what I'm getting on my laptop. The numbers varied somewhat, but the relative order stayed consistent.

AES-128-CBC 23,371KB/sec – 53,847KB/sec (depending on block size)
AES-192-CBC 21,672KB/sec – 46,580KB/sec
AES-256-CBC 20,288KB/sec – 38,053KB/sec
DES-CBC 22,818KB/sec – 24,282KB/sec
DES-EDE 8,726KB/sec – 8,738KB/sec

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 csil-linux 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.

.