

Name:

## Information Assurance: Homework 3

Due September 19, 2007 on compass. Late homeworks will only be accepted until September 21, 2007 to ensure that answer keys can be posted before the exam.

1. This portion of the homework involves working with AES and DES encryption. Pull the AES reference library from <http://www.cs.uiuc.edu/class/fa07/cs461/aes-files.zip>. This library includes a test AES program which you can augment as necessary. It also includes a makeKey program which creates a key of the specified length using the rand() pseudo-random function. Compile it for your target system. I have tested this program on Windows XP using cygwin and MSDev and on Linux. Makefiles and project files are included. Everyone should have access to the csil-linux\*.cs.uiuc.edu and the dcllnx\*.ews.uiuc.edu machines.
  - a. Fetch an encrypted file and a key file from <http://www.cs.uiuc.edu/class/fa07/cs461/hw3-enc.bin> and <http://www.cs.uiuc.edu/class/fa07/cs461/key07-192>. Decrypt the file using ECB mode. It should result in a plain English file. Submit the resulting plaintext.
  - b. Select a file to encrypt using a key in CBC mode. Submit the encrypted file, key file, and initialization vector file. Indicate the size of the key.
  - c. Try encrypting your file in ECB mode using different key lengths: 128 bit, 192 bit, and 256 bit keys. Time the performance difference. AES operations are very fast. You will want to use a high resolution timer such as the gettimeofday or clock\_gettime system calls. In addition, you will probably need to perform your target measured operation multiple times to have something that can be measured by the clock granularity. Plus, the I/O time of the program is likely to be much bigger than the actual encryption time. You will need to insert your timer calls appropriately to avoid measuring I/O. Submit a table of the key length and the associated average encrypt time.
  - d. Repeat part c using DES with its 56 bit key. On Linux the function ecb\_crypt and des\_setparity should do the trick. On Windows the base provider for the Crypto API should provide a DES encryption operation. Add a row to your table from part c showing the average time for the DES encryption operation.
2. Consider the Diffie-Hellman key exchange algorithm. This public key algorithm is based on the hard problem of computing discrete logarithms. Assume an efficient solution for calculating a discrete logarithm was found. How would this affect the strength of the Diffie-Hellman algorithm? Could it be attacked? How?
3. Work with Gnu Privacy Guard (GPG). You can access GPG from <http://gnupg.org>. I have used this on Linux and installed it via yum on my personal system. It should already be installed on the University Linux systems.

Name:

Type “man gpg” to check if it is installed. Once you get your GPG system operational perform the following tasks:

- a. Create a key pair.
  - b. Get your key signed by at least one other person. Submit an exported version of your signed public key.
  - c. Encrypt a file using the instructor’s public key (at <http://www.network-geographics.com/home/shinrich/skh-pubkey.asc> with fingerprint 388E 7466 4DD3 390E 8F36 A535 474D 5DC9 4912 BF7E). and sign it with your key. Submit the signed and encrypted file.
4. Alice wants to switch a document on Bob. Bob has seen the original document and is willing to sign a crypto hash of that document. They are using a 32 bit DES-CBC hash.
- a. How many variations of the new document will Alice need to try on average to find another document with the exact same crypto hash as the document Bob is willing to sign?
  - b. Alice is willing to make changes to the original document too (by adding spaces or other changes that don't change the meaning of the document). How many document comparisons will Alice need to make on average to find an original document variant and a variant of the new document with the same crypto hash.