

Name:

Information Assurance: Homework 3 Answer Key

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.

Not many comments on common errors until we finish grading. I'll update another version of the comments then. In the meantime here are comments on what we expect to find.

- 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.

See the posted plaintext excerpt from Heidi.

- 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.

We should have received a key file, an encrypted file and in IV file.

- 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.

Name:

We would expect to see that the performance slows as the key length increases. AES runs more rounds to incorporate more keying material. For the size of the files you are testing, the performance difference is negligible. But it is good to see the difference and it is important to know the rate difference when designing a system. The network bandwidth requirements may dictate an upper bound on the key length.

- 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.

I would expect to see the DES performance be worse than the AES performance. Certainly this is the case for triple DES which would give you similar crypto security to the AES-128. Your mileage may vary though based on the particular DES implementation your system has.

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?

Since Diffie-Hellman's strength is based on the difficulty of the discrete logarithm problem, finding an efficient solution for that problem will eliminate the cryptographic strength of Diffie-Hellman.

Consider Alice and Bob with p , g , and private keys k_A and k_B . They will exchange public keys:

$$KA = g^{k_A} \text{ mod } p$$
$$KB = g^{k_B} \text{ mod } p$$

Eve intercepts those messages. The values of p and g are public, so she uses the new discrete logarithm algorithm to solve for k_B and k_A . Then she can compute the shared value directly.

$$K_{\text{shared}} = g^{k_A k_B} \text{ mod } p$$

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

Again, I'll provide more comments after grading.

- a. Create a key pair.

Name:

- 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.

As noted in the news group, the size of the DES-CBC hash would be the block size of DES which is 64 bits. I make a typo writing up the question thinking ahead to the answer.

The intent of this question was to explore the difference in cost of finding a target document and a replacement document with the same hash file. In the first case, we take the naïve approach of fixing the target document and search until we find a replacement that matches. In the second case we alter both documents and leverage the birthday attack.

- 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?

This the same problem having a bag with n marbles. One is white and the other $n-1$ are black. You have people pull a marble from the bag and put it back. How many people have to select until you have a 50% probability that someone pulled a white marble from the bag. Since a good crypto hash has nearly random output, we can make incremental changes to the replacement document and have a close to uniform random spread of crypto values.

So first calculate what is the probably of having k people and none of them pull the white marble (or k documents and none of them match the target crypto hash).

$$P(\text{none of } k \text{ get the target}) = ((n-1)/n)^k$$

$$P(\text{at least one of } k \text{ gets the target}) = 1 - P(\text{none of the } k \text{ get the target}).$$

Once this probability is less than $1/2$, the inverse (the probability that at least one of k gets the target) will be greater than $1/2$.

$$P(\text{none of the } k \text{ get the target}) = ((n-1)/n)^k < 1/2.$$

$$2 * (n-1)^k < n^k$$

$$\lg_2(2) + k * \lg_2(n-1) < k * \lg_2(n)$$

Name:

$$n = 2^{32}$$

$$1 + k \cdot \lg_2(2^{32}-1) < 32k$$

$$1/(32 - \lg_2(2^{32}-1)) < k$$

2,977,044,474 < k (plus or minus a few, there was some variance between the calculators we used)

Another way of thinking about this is in terms of expected value rather than probability.

Expected number of trials = $1 \cdot P(\text{find match in first trial}) + 2 \cdot P(\text{find match in second trial}) + 3 \cdot P(\text{find match in third trial}) + \dots = 1 \cdot 1/n + 2 \cdot 1/n \cdot (n-1)/n + \dots$

$$= \sum_{i=1 \text{ to infinity}} i \cdot ((n-1)/n)^{i-1} \cdot 1/n$$

If you take the limit you get a value that is within a factor of two of the probability method. Either way is fine for this homework. The key thing to note that the value in part a is much bigger than the value in part b.

- 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.

In this case, we also consider the probability that in k people we don't have a matching birthday. This is equivalent to thinking about a bag of n black marbles. Each person picking replaces with a white marble.

$$P(\text{no one picks a white marble in } k \text{ people}) = \text{Prod}(i = 1 \text{ to } k) (n-i)/n = n!/((n-k)! \cdot n^k)$$

We need to solve for k so this probability is less than $1/2$. However, there is no nice closed form for this. The wikipedia page presents some bounds.

From the wikipedia article we get the bound (letters translated to match what we have been using in this writeup).

$$k(p, n) = \text{sqrt}(2 \cdot n \cdot \ln(1/(1-p)))$$

$$k(.5, 2^{32}) = \text{sqrt}(2 \cdot 2^{32} \cdot \ln(2)) = 77,162$$

This is close to the $2^{(m/2)}$ discussed in class. $2^{16} = 65,536$. The wikipedia article on the birthday attack (http://en.wikipedia.org/wiki/Birthday_attack) also gives a more specific bound of $1.17 \cdot 2^{(m/2)}$. $1.17 \cdot 2^{16} = 76,677$.

Then for this to work out for a match between two groups of message variants, you would need to create one group of that size for the Alice desired message and another group that size for the Bob desired message. So this adds another factor of two for the number of message hashes you would need to compute.

Name:

I assume you would make two hash maps one for each message type. As you compute a message variant, you put it in the hash map keyed by the crypto hash value. Then you look in the other hash map to see if it already has an entry for your crypto hash value.

In either case, these bounds are much, much less than the result for part a. In the case of the 32 bit hash, they are very attackable.