Homework 1

CS161 Computer Security, Fall 2008
Assigned 9/15/08
Due 9/24/08

For your solutions you can submit a hard copy; either hand written pages stapled together or a print out of a typeset document[1]. Question 2 is a small programming assignment; for that question you should also electronically submit files according to instructions which will be posted on the website later this week.

As explained in the first lecture, the homeworks are to be solved and written up individually. Students are permitted to discuss the homework in order to clarify questions, but should not share solutions or otherwise collaborate. Also, students should not attempt to search the Internet for solutions. If any sources other than the lectures or official textbooks are used, they must be clearly cited.

If any of the questions is unclear or you suspect there is a mistake in the homework, please post a message on the newsgroup about it!

1 Encryption

1. (1 point) Which of the following security goal(s) does encryption address : (1) Confidentiality (2) Integrity (3) Sender authentication (4) Non-repudiation.

2. (1 point) A one-time pad is provably secure. What makes it hard to use in practice?

3. (1 point) Suppose you obtain two ciphertext C, C' encrypted using one-time pad, with key K and its bitwise complement $\bar{K}$ respectively. What can you infer about the corresponding plaintext messages?

4. (1 points) What main advantage does public-key encryption offer over secret-key encryption.

2 Block Ciphers

This question contains a small programming assignment that is designed to ensure that your named UNIX accounts are properly set up for CS161 while also illustrating some properties of block ciphers.

We've published a code skeleton with a number of BMP image files[2] at :

`http://inst.eecs.berkeley.edu/~cs161/fa08/Homeworks/hw1-program.tar.bz`.

Untar the file by running `gtar jxf hw1-program.tar.bz`, and run `make` to compile the program. Then type

[1] LaTeX is the most suitable tool for typesetting mathematical documents

[2] From `http://xkcd.com/`.

```
./encrypt plaintext-medium.bmp out.bmp
```

to to attempt to encrypt the image. The skeleton will pass a null pointer into `write()` and fail with a "bad address" error until you fill in the solution to the assignment.

We've tested the code on Linux and on `cory.eecs.berekeley.edu`. We will use `cory.eecs.berkeley.edu` to grade the assignment, so make sure that your code builds and runs on that machine.

The two bitmap files contain uncompressed versions of the same image. One contains a monochrome (1 bit per pixel) version of the picture, and the other contains a 200% scaled RGB (24 bits per pixel) version of the picture.

When you've completed the question, submit the files `encrypt_ecb.c`, `plain.bmp`, `crypt.bmp`, and `encrypt_cbc.c`. Instructions for electronic submissions will be given on the website later this week.

1. (3 points) Edit `encrypt.c` so that it implements DES encryption in electronic codebook (ECB) mode. This files should be submitted as `encrypt_ecb.c`. Encrypt the two included image files.

2. (3 points) The skeleton intentionally avoids encrypting the BMP file's header so you can use standard software to inspect the encrypted data. Open the two encrypted files in your favorite image editor. (We used the GIMP.) Was the encryption effective? Explain the difference between the two encrypted files.

3. (3 points) Find a small (under 1MB uncompressed) image file that would be more effectively encrypted using a block cipher in ECB mode. Explain your choice of image. What sorts of attacks is this scheme vulnerable to?

   Convert the image to an uncompressed (not RLE encoded) 24-bit BMP format and encrypt it. Name the files `plain.bmp` and `crypt.bmp` and submit them.

4. (4 points) Edit `encrypt.c` so that it implements DES encryption in cipher block chaining (CBC) mode. This files should be submitted as `encrypt_cbc.c`. For this exercise, a block of all 0's may be used as the initialization vector. Encrypt the two original image files and view the result.

   How do the results differ when CBC is used instead of ECB? Briefly state why that is the case.

3 RSA Public-Key Encryption

1. (5 points) Alice and Bob are using public keys (e_1, N_1), (e_2, N_2) respectively. Suppose you are informed that their RSA moduli N_1, N_2 are not relatively prime. How would you break the security of their subsequent communication? It is sufficient to show that you can get $\phi(N_1)$ and $\phi(N_2)$.

2. (5 points) Suppose that a system uses textbook RSA encryption. An attacker wants to decrypt a ciphertext c to obtain the corresponding confidential plaintext m. Assume that the victim system readily decrypts arbitrary ciphertexts that the attacker can choose, except for ciphertext c itself. Show that the attacker can obtain m from c even under this setting, i.e a chosen ciphertext attack is possible.

3. (**Extra Credit**, 5 points). Show that an attacker who discovers the private key (d, N) for a public key (e=3,N), can efficiently factor $N = p.q$. Recall, we compute $ed = 1(mod\ \phi(N))$, such that $d < \phi(N)$.

4 Hash functions and Collisions

You are designing a multi-user OS. In your OS, users log into their respective accounts using passwords. It is dangerous to store user passwords in a file on the computer, because someone who obtains the file gets access to all passwords. As a solution, you decide to store the username and corresponding password's hash value in the file called `hpasswd`. Assume that you use an idealized, perfectly random hash function $h(x)$, i.e h is selected uniformly at random from all functions mapping $\{0,1\}^*$ to $\{0,1\}^k$. As always, h is publicly known. When a user logs in with password p, the OS grants access to the user iff $h(p)$ matches the entry for that user in `hpasswd`. Assume $k = 20$ in your OS.

There are some weaknesses in this mechanism of your OS. A collision in the password hash of 2 users, allow them to log in as each other.

1. (5 points) Suppose an attacker (a normal user) wishes to log in as the system administrator (the superuser) by trying random passwords (without repeating a guess he has already tried). What is the minimum number of password guesses that the attacker has to try to have a success probability greater than 0.6%.

2. (5 points) You want to limit is the probability of user password collisions, to below 20% in your design. That is, the probability of *any* two users' password hashes matching should be below 20%. What is the maximum number of users (N) you should allow in your OS?

Hint: To solve the above two subparts of this problem, you may need to use an approximation, as direct computation of k may be difficult with a fixed precision calculator. For this question it is acceptable to look up and cite outside sources for any useful approximations. Your answers may be approximate, importance will be given to steps shown for arriving at the solutions.

5 MAC and Digital Signatures

1. (1 point) What is non-repudiation?

2. (7 points) Suppose Alice has to sign a contract with Mallory using an RSA signature where, the signature is computed by $s = (h(m))^d\ mod\ N$ (d is Alice's private key). Assume that the hash function is an idealized, perfectly random hash function $h(x)$, i.e h is selected uniformly at random from all functions mapping $\{0,1\}^*$ to $\{0,1\}^k$, and h is publicly known. Mallory has managed to find 40 distinct places where he can make a slight change in the contract: adding a space at the end of line, adding a comma, replacing with equivalent words (like replacing "agreed to pay" with "is obliged to pay"), etc. Surely, Alice does not object any such minor change in the contract and is willing to sign a contract with any of these minor changes.

    - (2 points) How many possible versions of the contract can Mallory generate that Alice would be willing to sign?

- (3 points) Mallory creates a fraudulent contract reflecting a substantial increase in amount that Alice owes to Mallory. He computes the hash of the fraudulent contract as $h(f)$ and finds a version of the good contract that hashes to the same value $h(f)$. What is a minimum safe output size for the hash function (i.e k =?) to make these collisions unlikely? (An approximate answer with appropriate reasoning is acceptable. You need not show any calculations to get 'k'.)
- (2 point) What can Mallory do by finding such a collision, to force Alice to pay an increased amount in court?

3. (5 points) Recall from class that in an RSA signature scheme, the signature is computed by $s = (h(m))^d mod\ N$, where d is the private key. Consider an naive revised scheme where the signature is computed as $s' = m^d mod\ N$ instead. Show that it is possible to forge signatures for some messages in the latter (revised) scheme.

   Hint : Suppose Alice uses the latter (revised) scheme to sign two contracts m_1, m_2, generating signatures s'_1, s'_2 respectively. Construct an attack from s'_1, s'_2.