Homework 1

CS161 Computer Security, Spring 2008
Assigned 2/4/08
Due 2/13/08

This homework assignment is due Wednesday, February 13 at the beginning of lecture. Please bring a hard copy to class; either hand written pages stapled together or a print out of a typeset document[1] is fine. 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 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. **One-Time Pads**

   To communicate using a one-time pad, Alice and Bob would first need to share a secret, random ℓ-bit string $r \in \{0,1\}^\ell$ (e.g., established at an in person meeting). Then given their shared secret r, Alice could later privately send an ℓ-bit message $m \in \{0,1\}^\ell$ by sending Bob $s = m \oplus r$. Bob could then in turn compute the message as $s \oplus r = m$.

[1] LaTeX is the most suitable tool for typesetting mathematical documents. A good tutorial for those interested in learning to use it to prepare their homeworks may be found at `http://tinyurl.com/3bvag4` .

Having just learned about the one-time pad, Alice is excited about the security it offers. However, she is concerned about the inconvenience of establishing the shared key r with Bob. The following protocol occurs to her.

> When Alice is ready to send her message m, she randomly selects $r_1 \in \{0,1\}^\ell$ and sends Bob $s_1 = m \oplus r_1$. Bob then randomly selects $r_2 \in \{0,1\}^\ell$ and sends Alice $s_2 = s_1 \oplus r_2$. Next, Alice computes $s_3 = s_2 \oplus r_1$ and sends it to Bob. Bob may then compute the message as $s_3 \oplus r_2 = m$.

This idea seems similar to the one-time pad, but does not require prior distribution of a shared key.

(a) (1 point) Is Alice's protocol secure?

(b) (2 points) If your answer is no, give an attack that breaks the protocol. If your answer is yes, briefly specify your attack model and state why you think the scheme might be secure under that model (no formal proof is necessary).

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/sp08/hw01code.tar.gz`. Untar the file by running `gtar zxf hw01code.tar.gz`, and run `make` to compile the program. Then type

```
./encrypt_ecb 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.

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

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.

(a) (3 points) Edit `encrypt_ecb.c` so that it implements DES encryption in electronic codebook (ECB) mode. Encrypt the two included image files.

(b) (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.

(c) (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.

(d) (3 points) Edit `encrypt_cbc.c` so that it implements DES encryption in cipher block chaining (CBC) mode. 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.

(e) (extra credit, 4 points) Alice and Bob decided to use a modified version of the homework submission for secure communications. Alice tried to send her first encrypted email using one of the encryption programs:

    From: Alice
    To: Bob
    Date: Feb 5th, 2008 12:34:56PM

Bob,

I just generated a key. Did it work?

-A

However, Mallory intercepted it before it reached Bob's computer. A few seconds later (far too soon to brute-force a 56-bit key), Mallory encrypted a completely different message using Alice's key, and sent it to Bob. Eve saw the forged message, and now everyone's upset. Mallory told you there's a bug in the implementations of `main()` that we provided to you. What happened?

3. **Hash Functions and Collisions**

Let k be a positive integer and let $h : \{0,1\}^* \to \{0,1\}^k$. For this problem we will assume h is an idealized, perfectly random hash function. Specifically, assume h is selected uniformly at random from all functions mapping $\{0,1\}^*$ to $\{0,1\}^k$.

An attacker is interested finding collisions of h. Assume the attacker treats the hash function as a black box; i.e., the only operation they can perform is to compute the hash function on an input and observe the result. Further assume they can perform at most one million hash computations per second.

Hint: to solve the following two problems, 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. Briefly show how you arrived at your answers.

(a) (6 points) Suppose the attacker is interested in finding a preimage $x \in \{0,1\}^*$ which hashes to a particular value $y \in \{0,1\}^k$. What is the minimum value of k that ensures the attacker will have at most a 0.1% chance of succeeding in one year?

(b) (6 points) Suppose the attacker is interested in finding any collision, that is, preimages $x_1, x_2 \in \{0,1\}^*$ which both hash to the same value in $\{0,1\}^k$. What is the minimum value of k

that ensures the attacker will have at most an 80% chance of succeeding in one year?

4. **ElGamal and Chosen Ciphertext Attacks**

   Recall that to use the ElGamal public key encryption scheme, Alice randomly selects a private key $x \in \mathbb{Z}_p$ and computes her public key as $y = g^x \mod p$, where g is a publicly known generator of $\mathbb{Z}_p^*$. To encrypt a message $m \in \mathbb{Z}_p^*$ for Alice, Bob randomly selects $r \in \mathbb{Z}_p$ and computes the ciphertext as $c = (g^r, m \cdot y^r)$.

   (a) (3 points) Assume you are given an ElGamal public key y (but not the private key). Assume ciphertexts $c_a = (c_{a_1}, c_{a_2})$ and $c_b = (c_{b_1}, c_{b_2})$ are encryptions of some unknown messages m_a and m_b.

   Show how you can construct a ciphertext which is a valid ElGamal encryption of the message $m_a \cdot m_b \mod p$.

   (b) (4 points) Show how the above property of ElGamal leads to a chosen ciphertext attack.

   That is, assume you are given an ElGamal public key y and a ciphertext $c = (c_1, c_2)$ which is an encryption of some unknown message m and that you are furthermore given access to an oracle that will decrypt any ciphertext other than c. Based on these things, compute m.

5. **Factoring and More**

   Given an integer n, we say that m is a *non-trivial factor* of n if $m|n$, $m \neq n$, and $m \neq 1$.

   (a) (4 points) Assume $n = pq$, where p and q are prime. It can be shown that for every $x \in \mathbb{Z}_n^*$, there exists a $y \in \mathbb{Z}_n^*$ such that $y \neq x$, $y \neq -x$, and $x^2 \equiv y^2 \mod n$.

   Assume you are given such an x and y in $\mathbb{Z}_n^*$, that is, $x^2 \equiv y^2 \mod n$, $x \neq y$, and $x \neq -y$. Show that $\gcd(x - y, n)$ is a non-trivial factor of n.

   (b) (1 point) Suppose we are considering a function $h : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ for use as a hash function, where $h(x) = x^2 \mod n$. Does h satisfy the compression property of a hash function? No justification of your answer is required.

(c) (8 points) If we assume the difficulty of factoring, does h satisfy the preimage resistance (a.k.a. one-way) property of hash functions?

If your answer is no, give a probabilistic polynomial time algorithm that, when given n and y, will output an $x \in \mathbb{Z}_n^*$ such that $h(x) = y$ or abort if no such x exists.

If your answer is yes, show how such an algorithm could be used to factor n in expected polynomial time.

(d) (3 points) If we assume the difficulty of factoring, does h satisfy the second preimage resistance (a.k.a. weak collision resistance) property of hash functions?

If your answer is no, give a probabilistic polynomial time algorithm that, when given n and x, will output an $x' \neq x$ such that $h(x) = h(x')$ or abort if no such x exists.

If your answer is yes, show how such an algorithm could be used to factor n in expected polynomial time.