Problem Set 1

This problem set is due *via email,* to `6.857-submit@theory.csail.mit.edu` on *Monday, September 18* by the beginning of class.

You are to work on this problem set in groups of three or four people. Problems turned in by individuals, pairs, pentuples, etc. will not be accepted. Be sure that all group members can explain the solutions. See Handout 1 (*Course Information*) for our policy on collaboration. If you do not have a group, let us know.

Homework must be submitted electronically! Each problem answer must appear on a separate page. Mark the top of each page with your group member names, the course number (6.857), the problem set number and question, and the date. We have provided templates for LaTeX and Microsoft Word on the course website (see the *Resources* page).

Grading and Late Policy: Each problem is worth 10 points. Late homework will not be accepted without prior approval. Homework should not be submitted by email except with prior approval. (*Somebody* from your group should be in class on the day that the homework is due.)

With the authors' permission, we will distribute our favorite solution to each problem as the "official" solution—this is your chance to become famous! If you do not wish for your homework to be used as an official solution, or if you wish that it only be used anonymously, please note this on your homework.

Problem 1-1. Security Policy.

The California legislature has recently passed the Identity Information Protection Act, which requires that state-issued IDs that contain remotely-readable RFID chips must contain adequate security features to prevent them from being read by unauthorized parties. RFID chips are designed to store unique identifiers that will be broadcast in response to a particular radio signal.

The bill was motivated by concerns that these IDs might be remotely read without the user's knowledge, revealing personal information that could be used to commit fraud, identity theft, or gain unauthorized access. To read more about the bill, its intent, and its motivations, got to the *Resources* page on the course web site and click on *California RFID Bill Nears Approval.*

Your Task For this bill to be properly enforced the state must develop a security policy for RFID IDs. You are to help California lawmakers by writing your own short security policy for this scenario. Specifically, write a security policy for a state-issued ID card that includes an RFID meant to remotely communicate a unique identifier. The ID cards are to be used to facilitate efficient identification at pre-specified locations, including: airports, government services offices, and the Mexican border. Your security policy should take into account the concerns of the lawmakers (i.e., the IDs being remotely read without the user's knowledge leading to identity theft), but also not be too restrictive.

For help on writing a security policy go to the *Resources* page on the course web site and click on *Sample Solutions from PS1 2003.* See question 1-4, which asked students to develop a security policy for either the MIT Card or Apple's iPod. Sample solutions for both, as well as a short discussion from the TAs regarding common omissions, are included. These should help guide you in terms of content, format, and length.

Problem 1-2. Remote Voting

Ned Nerdle has been thinking about the problem of "remote voting" (i.e. absentee voting, voting over the phone or over the Internet), and has come up with the following proposal.

1. When each voter registers to vote, he/she must establish a 5-digit secret PIN with the election authorities in his/her jurisdiction. This PIN is known only to the voter and to the election authorities. The voter is encouraged not to write his PIN down, but to memorize it. The voter can return to the town hall at any time if he wishes to change his PIN.

2. The PIN is used to establish a voter-specific “secret code” used for voting. Each element of the code is also a 5-digit number, like the PIN. An element of the code represents “YES” if it agrees with the PIN in EXACTLY one position; otherwise it represents “NO”. Thus, there are

   $$5 * 9^4 = 32,805$$

   possible ways of saying “YES” and

   $$100,000 - 32,805 = 67,195$$

   possible ways of saying “NO”.
3. When the voter votes, he uses random “YES” and “NO” codes to represent his votes. These may be written on the paper absentee ballot, entered on a phone keypad after each candidate’s name is read (if voting by phone), or entered in a browser form (if voting over the Internet).

For example, if Alice’s PIN is “31415”, then her filled-in ballot:

George Shrub:	27408
John Carie:	82841
Rolf Nadir:	31980

represents a vote for George Shrub.

Your Task Your assignment in this problem is to evaluate Nerdle’s proposed voting scheme. What might Nerdle be trying to accomplish? What pros (if any) or cons (if any) do you see for this scheme, with respect to security or other criteria, compared to existing remote voting schemes? Can you suggest modifications or improvements to this proposal?

Problem 1-3. MD5 Collisions In 2005, Wang and Yu published an attack on the MD5 hash function. Specifically, they discovered a technique that can efficiently produce collisions. See `http://www.infosec.sdu.edu.cn/paper/md5-attack.pdf` for the paper. The ability to produce collisions can introduce severe security vulnerabilities into schemes which depend on MD5 hash values as an indicator of integrity. To highlight the power of Wang and Yu’s attack, Daum and Lucks produced two postscript files with identical MD5 hashes. When read, however, one is a letter of recommendation and the other is a security clearance!

To see for yourself, download *letter_of_rec.ps* and *order.ps* from the *Resources* page of the course web site. View each file, then calculate their MD5 hash. Notice, we have made a Windows and Linux version of an MD5 hash calculator available under *Resources*. The linux version, *md5sum*, is included in many distributions, so you may not need to download it.

Familiarize yourself with Daum and Lucks attack by reading the “How it Works” section of `http://www.mathstat.dal.ca/~selinger/md5collision/`, as well as the description on `http://www.cits.rub.de/MD5Collisions/`. You should then view the source code of the postscript files. You’ll notice that the core programming insight behind the attack is the following postscript structure:

```
(v1)(v2)eq{c1}{c2}ifelse
```

Which evaluates the commands $c1$ if $v1 = v2$ and instead evaluates $c2$ if $v1 \neq v2$.

Your Task In this problem we ask you to extend the ideas behind Daum and Lucks sneaky files to produce four different postscript files that each produce the same value when hashed with MD5. Specifically, each file, when viewed with a postscript viewer, should display a unique output. For example, the four files can display: 00, 01, 10, and 11, respectively, or four unique orders, etc. Each group should attach their four files to their e-mail submission of this problem set. The TAs will verify that each file displays something different yet all hash to the same value.

To calculate MD5 collisions we recommend that you use Marc Steven's Fast MD5 Collision Generator, which is available under the *Resources* section at the course web site and at `http://www.win.tue.nl/hashclash/`. Unfortunately, this program is only available as a Win32 exe, so you will have to run it on a Windows machine. If you have trouble finding access to such a machine, contact the TAs.

Run the program from a Windows command line as follows:

```
fastcoll_v1.0.0.1.exe -p <prefix_file>
```

Where `<prefix_file>` is an arbitrary text file. The collision generator will then output two files: msg1.bin and msg1.bin. Each includes `<prefix_file>` followed by a unique pattern of 128 bytes. Both msg1.bin and msg2.bin will hash to the same value. Note, if `<prefix_file>`is not a multiple of 64 bytes, the program will first pad the file as needed to obtain this property.

A few words of warning:

- If the output of the collision generator includes unmatched parens (when the values are interpreted as ASCII characters, i.e., when viewed in a text editor), this will confuse the postscript processor. If there are unmatched "("'s, then you can simply add extra ")"'s at the end of the output. If there are unmatched ")"'s, you will have to try again and generate a new output.
- Using Windows' Notepad to edit the text files can be problematic. We have observed that Notepad will occasionally cause two files to hash differently even though the text changes to each were identical. The easiest way to avoid this occasional problem is to edit the files in emacs on a Linux machine or under cygwin—which seems to work consistently.
- Finally, we expect each group to generate their own collisions. No two groups' files should include the same hash blocks.