Massachusetts Institute of Technology
6.857: Network and Computer Security
Professor Ronald L. Rivest

Handout 9
September 25, 2003

Problem Set 3

This problem set is due *on paper,* in room 6-120 on *Thursday, October 2* at 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, seek partners by emailing `6.857-students-public@mit.edu`.

Homework must be typed! Each problem answer must appear on separate sheets of paper. Mark the top of each sheet with your name(s), the course number (6.857), the problem set number and question, and the date. **Homework must be typed and clear.** We have provided templates for LaTeX and Microsoft Word on the course website.

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 3-1. Prime Triangles. A triple of numbers (a, b, c) is called a *k-digit prime triangle* if:

- a, b, and c are k-digit integers[1], and
- each concatenation $a \circ b$, $b \circ a$, $a \circ c$, $c \circ a$, $b \circ c$, $c \circ b$ is a prime number. (Formally, if x and y are k-digit integers, $x \circ y = 10^k x + y$.)

For example, the numbers $(1, 3, 7)$ form a 1-digit prime triangle because 13, 31, 17, 71, 37, and 73 are all prime numbers.

Find a k-digit prime triangle for the largest k you can. The search for this value of k should take no more than two hours of processing time on a "reasonable" personal computer. In addition, you should *not* use the library version of any primality-testing procedure—instead, write your own (you may use big-integer libraries for all other mathematical operations). Describe your approach to the problem and how your algorithm works.

Problem 3-2. Block Ciphers. There are two ways of implementing a cryptographic file system. The first approach encrypts each file as it is written and decrypts the file as it is read. One advantage of this approach is that it can be implemented without modifying an operating system using hooks that are present for implementing remote file systems. One disadvantage of this approach is that it cannot be implemented in hardware.

Design the cryptographic aspects of a hardware-based disk encryption system. Your device will fit between a hard drive and a computer. Data written to the hard drive should automatically be encrypted, while data read back should be decrypted. Use a block cipher as your encryption function. Be sure to answer these questions:

- What encryption algorithm would you use, and why?
- In what mode of operation should you run the block cipher? If the mode requires an initialization vector, how will it be set?
- How hard is it for the user to change her key? Can you design a system that allows the user to change her key very quickly?

[1]an integer x has k digits if $10^{k-1} \leq x < 10^k$

- How should the user provide her key to the system?
- What are the advantages and disadvantages of a hardware-based disk encryption system of this type?

Problem 3-3. MAC Attack. Consider the following message authentication code using the block cipher AES: there is a large, publicly-known prime $p = 2^{128} - 159$. Alice and Bob share a secret key (K, a), where K is a random 128-bit AES key, and a is a randomly chosen element from $\mathbf{Z}_p$.

To authenticate a message M, Alice first pads M in the following way: she unconditionally appends a 1 bit, followed by enough 0 bits to make the length of M a multiple of 128. She then breaks M into blocks $M_0, M_1, \ldots, M_{n-1}$ (each of length 128), and encrypts each block as in ECB mode: $C_i = \mathrm{AES}_K(M_i)$. The MAC for the message is:

$$\mathrm{MAC}_{(K,a)}(M) = \sum_{i=0}^{n-1} C_i a^i \pmod{p}$$

As an adversary, you have the following power: you can repeatedly query a black-box (which contains the secret key) on any message, and get back the MAC for that message. Your task is to compute a forgery (i.e., a valid ⟨message, MAC⟩ pair) for some *new* message that is different from all your queries. Describe how to successfully attack this MAC. Try to use as few black-box queries as possible.