Massachusetts Institute of Technology | Handout 3
6.857: Network and Computer Security | February 19, 2009
Professors Srini Devadas and Nickolai Zeldovich | **Due:** March 1, 2009

Problem Set 2

This problem set is due *online,* at `https://sec.csail.mit.edu/` on *Monday, March 1* by the beginning of class. No late submissions will be accepted.

You are to work on this problem set with your assigned group of three or four people. You should have received an email with your group assignment for this problem set. If not, please email `6.857-tas@mit.edu`. Be sure that all group members can explain the solutions. See Handout 1 (*Course Information*) for our policy on collaboration.

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: Problem 1 is worth 10 points, and problem 2 is worth 30 points.

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 in your profile on the homework submission website.

Problem 2-1. MAC variants

CBC-MAC is a way of implementing a MAC using a block cipher in CBC mode. Specifically, the CBC-MAC of a message m with key k is the last ciphertext block when encrypting message m with the block cipher using key k and an IV of all zeroes. Unfortunately, CBC-MAC is not a secure authentication code for variable-length messages, when an adversary can obtain MAC values for different length messages.

One way we might try to craft a secure scheme for messages of variable length is to append the message length m (i.e., the number of block-cipher blocks) to the message M, and compute the CBC-MAC of the resulting string, $M||m$.

(a) This proposed CBC-MAC scheme is not secure. Give an attack against it. Assume you can perform a chosen-plaintext attack—that is, you can query the MAC value for any message of your choice. However, ultimately you must produce a valid MAC value for a message that you did not query the MAC value of.

Now consider an alternative construction of a MAC from a block cipher, called XOR-MAC. Assume that you have a secure block cipher F with a 64-bit block size. Given a message M, the sender breaks M into blocks $M_1, \ldots, M_\ell$, each of length 32 *bits* (for simplicity we assume that the length of M is always a multiple of 32). Let $\langle i \rangle$ denote the binary representation of i as a string of exactly 31 bits, and let $\alpha.\beta$ denote the concatenation of two binary strings α and β. To authenticate M using a secret key K, the sender does the following:

- Pick a random 63-bit binary string r.
- Set $t = \mathrm{F}_K(0.r) \oplus \mathrm{F}_K(1.\langle 1 \rangle.M_1) \oplus \cdots \oplus \mathrm{F}_K(1.\langle \ell \rangle.M_\ell)$.
- Set the MAC of M to be the pair (r, t).

The verification algorithm is similar to the above (we leave it to you to reconstruct).

(b) Observe the initial 0 and 1 bits prepended to each block, e.g. in $\mathrm{F}_K(0.r)$ and $\mathrm{F}_K(1.\langle 1 \rangle.M_1)$. Suppose we were to remove them, and instead use a 64-bit r. Suggest an attack on this modified scheme.

Problem 2-2. Ben's Block Cipher

Ben Bitdiddle proposes the following block cipher. Ben's cipher operates on 128-bit input blocks and produces 128-bit output blocks.

Let $I_n = 0, 1, ..., n-1$. A key K consists of three parts (p, S, q):

- A permutation p of I_{128}. (i.e., $p[i] \in I_{128}$ for all $i \in I_{128}$, and $p[i] \neq p[j]$ if $i \neq j$.)
- A invertible byte-substitution table S that maps I_{256} to itself one-to-one. That is, S maps every possible "input" byte to exactly one "output" byte. (Such a table is known as an S-box and is a favorite tool of cryptographers.)
- A second permutation q of I_{128}.

The block cipher E works as follows, on an input message bit vector $M[i]$, $i = 0, 1, ..., 127$:

1. The bits are permuted according to p: $R[i] = M[p[i]]$ for $i = 0, 1, ..., 127$.
2. The bits of R are grouped into bytes in $T_0, T_1, ..., T_{15}$.
3. Each byte T_i is replaced by $S[T[i]]$, yielding byte sequence $U_0, U_1, ..., U_{15}$
4. The bytes are interpreted as a bit sequence again; call this sequence $V[i]$ for i in I_{128}.
5. The bits are permuted according to q: $C[i] = V[q[i]]$ for i in I_{128}.
6. $C[0...127]$ is the output ciphertext.

(a) In Ben's scheme, there are many equivalent keys. These keys have different values for p, S, and q, but produce the same ciphertext for any given plaintext. Describe how to generate all keys equivalent to a given key (p, S, q).

(b) Describe a chosen-plaintext attack on Ben's cipher that recovers the unknown key (p, S, q) or an equivalent key.

To demonstrate that your attack works, you have write an implementation that can break a randomly generated key. We have set up a Web server that implements Ben's encryption algorithm at `http://bitdiddle-sec.heroku.com/`. Prove that your implementation works by getting your team's name on the list of successful teams.

To follow good security practices, we have exposed our Web server implementation at

`http://github.com/costan/bitdiddle_sec`

The server is powered by Heroku's shared hosting infrastructure, and is subject to their quotas for free accounts. Please don't issue denial of service attacks.