Massachusetts Institute of Technology
6.857: Network and Computer Security
Professors Ronald L. Rivest and Yael Tauman Kalai

Handout 3
February 18, 2020
Due: March 2, 2020

Problem Set 2

This problem set is due on *Monday, March 2, 2020* at **11:59 PM**. Please note our late submission penalty policy in the course information handout. Please submit your problem set, in PDF format, on Gradescope. *Each problem should be in a separate PDF.* When submitting the problem in Gradescope, ensure that **all your group members are listed on Gradescope**, and not in the PDF alone.

You are to work on this problem set in groups. For problem sets 1, 2, and 3, we will randomly assign the groups for the problem set. After problem set 3, you are to work on the following problem sets with groups of your choosing of size three or four. If you need help finding a group, try posting on Piazza or email `6.857-tas@mit.edu`. You don't have to tell us your group members, just make sure you indicate them on Gradescope. 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 be provided as a separate pdf. 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: All problems are worth 10 points.

With the authors' permission, we may 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 your homework submission.

Our department is collecting statistics on how much time students are spending on psets, etc. For each problem, please give your estimate of the number of person-hours your team spent on that problem.

Problem 2-1. Side-Channel attack on AES

An attacker can use "side-channel" information to obtain a secret AES key.

We set up a server to simulate this type of side-channel attack. Our server encrypts random strings using AES with a secret key k. As a side-channel vulnerability, there is a slight delay whenever the number of 1 bits exceeds the number of 0 bits in the XOR outputs during each encryption (i.e. counting only the outputs of the XOR's when some round key is XOR-ed with the current state). The time is measured by the server and is part of the data given by the server.

If you query `https://courses.csail.mit.edu/6.857/2020/6857_aes.php` you will receive a (16 bytes (128 bits) of plaintext, 16 bytes of ciphertext, float value of time in seconds) triples. You can batch queries by including a 'num' parameter in the query, e.g. `https://courses.csail.mit.edu/6.857/2020/6857_aes.php?num=100` gives you 100 lines of tuples. The maximum number of queries per batch is capped to 1500. While you can make requests to the server many times, try to store the triples generated and other relevant information so as not to overload the server with requests.

The plaintext and ciphertext are printed as space-separated byte values in decimal; and the plaintext, ciphertext, and time leak are comma-separated. The line is ended with a semicolon.

You may find useful the NIST AES specification in this problem: `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

We have provided a locally runnable implementation in the files `server.py` and `aes.py`. The `aes.py` code contains the side-channel leak. Feel free to play with it on your local machine. It is not necessary to solve the problem, and is just there for reference.

(a) Let X be a random variable that is the total number of heads in t independent coin-flips of a fair coin, and let Y be another independent random variable that counts the number of heads in t coin flips, and then adds one. Suppose t is known.

Suppose Z is either $\left\lfloor \frac{X}{t/2} \right\rfloor$ or $\left\lfloor \frac{Y}{t/2} \right\rfloor$, but you don't know which. How many draws of Z do you expect to need, in order to determine with reasonable reliability (for example, guesses correctly 99% of the time) whether Z is $\left\lfloor \frac{X}{t/2} \right\rfloor$ or $\left\lfloor \frac{Y}{t/2} \right\rfloor$? (Note: an approximate bound should be good enough here, just make sure to state your assumptions.)

(b) Describe an algorithm for recovering the AES key k given the AES side-channel information described above. (Hint: try using the result of part (a) on the first bit of the first round key. What is t?)

(c) Recover the secret key k. Submit any code you used.

(d) How many (plaintext, ciphertext) tuples did your attack require? (Note that even when your attack seems to recover almost all the keybits correctly, it may still get one or two key bits wrong, which will result in an incorrect decryption of a ciphertext. Please report the number of plaintexts for which your algorithm has a good chance of outputting all key bits correctly.)

Problem 2-2. Hash Functions

Let's review some hash function properties and prove some useful facts about them.

Recall that a hash function h: $\{0,1\}^n \to \{0,1\}^d$ is one-way if you are given a $y = h(x)$ for a random $x \in \{0,1\}^n$, it is computationally infeasible to find any $x' \in \{0,1\}^n$ such that $y = h(x')$.

Such a hash function h (or a family of such hash functions) is said to be collision resistant if given the description of h it is hard to compute two different inputs $x_1, x_2 \in \{0,1\}^n$ such that $h(x_1) = h(x_2)$.

(a) Suppose that $h_1 : \{0,1\}^n \to \{0,1\}^d$ is a collision resistant hash function. Does it imply that $h_2 : \{0,1\}^{n-d} \times \{0,1\}^n \to \{0,1\}^d$ is also collision resistant, where h_2 is defined by $h_2(x,y) = h_1(x||h_1(y))$, for $x \in \{0,1\}^{n-d}$ and $y \in \{0,1\}^n$? Provide a proof or a counter example.

(b) Suppose that $h_1 : \{0,1\}^n \to \{0,1\}^d$ is a collision resistant hash function. Does it imply that $h_2 : \{0,1\}^{2n} \to \{0,1\}^d$ is also collision resistant, where h_2 is defined by $h_2(x) = h_1(x_2||x_4||\ldots||x_{2n})$, for $x = (x_1, \ldots, x_{2n}) \in \{0,1\}^{2n}$? Provide a proof or a counter example.

Problem 2-3. Message Authentication Codes

In this problem we explore the CMAC construction and its variants, and explore general encryption and authentication mechanisms.

(a) Recall that CMAC uses a CBC mode of operation for encrypting the message and a fresh key for the last block. Then, the output is the last ciphertext block. In addition, IV is set to 0.

1. Show that CMAC is not secure if we use the same key for all blocks (including the last one).
2. Suppose we use two different keys with a different key for the last block, but we let the IV be random (i.e. IV can vary). Is the MAC secure now? Show why or why not.

(b) Consider the following MAC scheme: a message M is hashed and the resulting value $h(M)$ is passed through an ideal block cipher E_k. The result, $E_k(h(M))$ is used as the MAC. Under what properties of the hash function is this MAC scheme secure? Or is it insecure no matter which hash function is used?

(c) Suppose the *same* key is used for all the blocks (including the last one), and assume that all messages are of *fixed* length L (for simplicity, you can think of L as corresponding to two blocks in the CMAC scheme). Namely, the adversary has access to an oracle that generates MACs of messages of length L and the adversary breaks the scheme if they manage to produce a MAC of a new L-bit message. Sketch an argument of why the scheme is secure.