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

Handout 2
Feburary 12, 2018
Due: Feburary 26, 2018

Problem Set 1

This problem set is due on *Monday, Feburary 26, 2018* 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.* Have **one and only one group member** submit the finished problem writeups. Please title each PDF with the Kerberos of your group members as well as the problem set number and problem number (i.e. *kerberos1_kerberos2_kerberos3_pset4_problem1.pdf*).

You are to work on this problem set 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 1-1. Security Policy for Cameras

Photojournalists have requested that manufacturers of handheld digital cameras (like Nikon, Canon, and Olympus) add an encryption feature to their cameras. The photojournalists have long argued that, when they're out in the field collecting footage and documenting evidence, the police, military, and border agents in countries where they work can examine and search their devices, thus putting their lives at risk. For example, if a photojournalist took a picture that painted the local government in an unflattering light and a border agent found that image, the journalist might face legal or physical reprisal.

Describe a security policy for a camera. Who owns the photos, and who is allowed to decrypt them? How are they encrypted? Where is the key stored? Do you use a public key encryption scheme or a symmetric one? Are the photos digitally signed for authenticity? We refer to `https://en.wikipedia.org/wiki/Public-key_cryptography` for an exposition of public-key cryptography, and to `https://www-ee.stanford.edu/~hellman/publications/24.pdf` for the seminal paper on public-key cryptography by Diffie and Hellman.

The policies you come up with should address each of the security goals discussed in class, though focus on the one(s) that are most relevant for cameras.

Given time constraints and the complexity of the problem, we expect your solutions to be less than comprehensive. That being said, keep in mind that there an adversarial party may try to interact with the camera, to learn sensitive information, and to cause some undesirable result.

(This problem is a bit open-ended, but should give you excellent practice in writing a security policy. We have included sample solutions from similar questions in previous years on the course website.)

Problem 1-2. Re-Using a One-Time Pad

It is well known that re-using a "one-time pad" can be insecure. This problem explores this issue, with some variations.

In this problem all characters are represented as 8-bit bytes with the usual US-ASCII encoding (e.g."A" is encoded as 0x41). The bitwise exclusive-or of two bytes x and y is denoted $x \oplus y$.

Let $M = (m_1, m_2, \ldots, m_n)$ be a message, consisting of a sequence of n message bytes, to be encrypted. Let $P = (p_1, p_2, \ldots, p_n)$ denote a pad, consisting of a coresponding sequence of (randomly chosen) "pad bytes" (key bytes).

In the usual one-time pad, the sequence $C = (c_1, c_2, \ldots, c_n)$ of ciphertext bytes is obtained by xor-ing each message byte with the corresponding pad byte:

$$c_i = m_i \oplus p_i, \text{ for } i = 1 \ldots n \ .$$

When we talk about more than one message, we will denote the messages as $M_1, M_2, \ldots, M_k$ and the bytes of message M_j as m_{ji}, namely $M_j = (m_{j1}, \ldots, m_{jn})$; we'll use similar notation for the corresponding ciphertexts.

(a) Here are two 12-character English words encrypted with a "one-time pad". Decide whether they were encrypted with the same pad or with different pads. If they are different pads, then explain why they cannot be the same pad. If they are the same pad, then decrypt the ciphertexts.

```
a6 a5 6d f4 8c a0 fc 86 d6 1f 2f e9
ac b9 60 e1 94 a3 f2 93 d2 01 24 f5
```

(b) Ben Bitdiddle decides to fix this problem by making sure that you can't just "cancel" pad bytes by xor-ing the ciphertext bytes.

In his scheme g is a random-looking permutation of bytes. That is, g is a 1-1 function mapping $\{0,1\}^8$ to $\{0,1\}^8$. They can be represented as byte-valued array G of size 256. Ben chose this function in a random manner using dice; they indeed look "random" – there is no apparent structure. The array G is public and given in the file **gbox.txt**.

The sequence $C = (c_1, c_2, \ldots, c_n)$ of ciphertext bytes is obtained by

$$c_i = g(m_i \oplus c_{i-1}) \oplus p_i, \text{ for } i = 1 \ldots n$$

where $c_0 = 0$.

Argue that Ben's scheme is decryptable. As part of your answer, explain how the recipient decrypts a ciphertext with Ben's scheme.

(c) Argue that Ben's scheme is like the OTP, "one-time secure": An adversary who hears one ciphertext, but doesn't have any information about the pad used, learns nothing about the message.

(d) Ben is confident that he can now re-use his pad, since there is no apparent way that one can "cancel" the effect of the pad on the message to obtain the ciphertext. For example, xor-ing ciphertexts doesn't seem to do anything useful for an adversary. So, he feels that he can now re-use a pad freely.

You are given the file **10ciphs.txt**, containing ten ciphertexts $C_1, C_2, \ldots, C_{10}$ produced by Ben, using the *same* pad P. You know that these messages are meaningful and contain only printable characters.

Submit the messages and the pad, along with a careful explanation of how you found them, and any code you used to help find the messages. The most important part is the explanation.

Problem 1-3. Spectre Usually, programs are modeled as black-boxes in cryptography. i.e., no information about the internal state of the program leaks during computation. Side-channel attacks work by exploiting this gap between the model and the real world. In the past, side-channel attacks have been used to subvert cryptographic primitives. Recently, two side-channel attacks called Spectre and Meltdown have been demonstrated that can potentially affect a broad class of programs. Your task in this question is to understand these vulnerabilities. We have included some 'readings'. Reading all of them is not mandatory.

1. A talk by Google Project Zero: `https://www.youtube.com/watch?v=6O8LTwVfTVs`.
2. A blog-post: `https://razorpay.com/blog/meltdown-paper-summary/`.
3. The papers detailing the attacks:[1] `https://spectreattack.com/spectre.pdf`, `https://meltdownattack.com/meltdown.pdf`.

(a) Describe how a timing attack works.

(b) Describe briefly how the Spectre attack works.

(c) Suppose Amazon is considering implementing one of the following proposed techniques to mitigating the Spectre vulnerability in AWS. For each proposal, explain (1) under what circumstances it would mitigate Spectre and (2) whether you would recommend Amazon pursue this option. Be sure to justify your answer to (2), taking into account the pros and cons of the proposed mitigation.

- Disable indirect branch prediction in all processors.
- Give individual processes the ability to disable indirect branch prediction when they are running.
- Given individual customers a choice of whether to disable indirect branch prediction for their virtual machines.
- Do nothing.

[1] There was a typo in the previous version where both the links were the same.