Problem Set 1

This problem set is due *online,* at `https://courses.csail.mit.edu/6.857/` on *Friday, February 21* by **11:59 PM**. Please note that 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: All problems are worth 10 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 1-1. Security Policy

The last few years have seen a sharp increase in smartphones and related devices (tablets, the iPod Touch, etc.) with the following properties:

- Applications are distributed through an "app store" or "market" operated by the OS vendor.
- There is only one local user of the device.
- Without a developer key (or exploitation of a local security vulnerability, aka "jailbreaking"/"rooting"), the end user can modify neither OS components nor, in many cases, applications.
- The device has somewhat limited resources in terms of power, storage, and bandwidth.

This distinguishes these devices from traditional PCs, where applications are usually obtained by the application developer and the end user can easily customize the OS or install applications. These changes allow several options for the device or OS manufacturer to avoid several security issues with traditional PCs.

Define a security policy for management of an app store for a new smartphone platform. Consider the lifecycle of an application, including how it is developed, uploaded to the app store, approved, downloaded and installed on devices, sold (if applicable), updated, and uninstalled. There are several parties involved, including the application developer, the app store, the device, and the user, who should approve various actions.

There are several issues you should address, including threats to confidentiality and integrity of data on the phone as well as availability of the phone itself, compromised application developers, dealing with security issues (bugs) in released apps, user privacy, backups of apps and data, and the like. You don't have to define a security policy for *running* applications on the device unless it's particularly relevant.

Problem 1-2. Couple-Time Pads

On the resources section of the class website, you can find twenty encrypted messages. We generated them by starting with twenty plaintexts and ten "one-time" pads, and picking a random pad to encrypt each plaintext. Therefore, each pad was used on average for two plaintexts.[1] Oops.

Your job is to identify which ciphertexts were encrypted with the same pad. Describe your approach and the groups you identified.

If it helps your analysis, the texts are taken from English Wikipedia articles about cryptography and are all about 1000 bytes long. The texts are all lower ASCII: letters and simple punctuation, but no accented characters, etc. The pads contain the ASCII characters A–Z, a–z, 0–9, +, and / selected randomly (we drew some number of random bytes and encoded them with base64). The encryption method is to XOR bytes of the plaintext message and the pad together. Each ciphertext is only as long as the corresponding plaintext, even if the pad was longer.

The encrypted messages and the script to generate them are on the class website.

Problem 1-3. Grøstl

Grøstl is a hash function submitted to the NIST's SHA-3 competition. In December, it was one of the five hash functions selected to remain in the third and final round of the competition; NIST will announce the winner in 2012[2].

Take a look at Grøstl's website at `http://www.groestl.info` and read at least sections 1–3 of its specification.

Grøstl has two fixed internal permutations: P and Q. They take an l-bit block and return an l-bit block, where l is 512 or 1024, depending on the output size. Here P and Q don't permute the l bits, they permute the input message space of 2^l possible blocks. Since P and Q are permutations, they are one-to-one functions, and hence invertible (and in fact are easily invertible).

It is not entirely clear from the documentation why the design for Grøstl has these particular choices for P and Q. For example, consider the following three "simplified" versions of Grøstl, which are identical to the submitted version except that in the three versions the following modifications are made:

1. Set P to the identity function $P(x) = x$
2. Set Q to the identity function $Q(x) = x$
3. Set $P = Q$

Try to figure out why the designers of Grøstl didn't use any of these variants, which would have been simpler and presumably faster. Are any of these modified hash functions unsuitable? Describe all the issues you find.

Note that we have found attacks of varying severity for each of these cases, all of which produce at least one collision. In particular, one of the cases is quickly seen to be seriously unsuitable, and we have also found at least one other second-preimage attack, i.e., for a message M we can produce another message M' such that $h(M) = h(M')$.

[1] During the Second World War and the Cold War, the Soviets ended up using one-time pads more than once, which allowed American codebreakers to slowly decrypt intercepted messages. Search the web for the "Venona project" for more information.

[2] The other finalists are listed on NIST's website at `http://www.nist.gov/hash-competition` .