

Problem Set 3

Submit this problem set in PostScript, PDF, or MS Word format to `6857-submit@csail.mit.edu` before lecture on the due date. We have provided templates for L^AT_EX and Microsoft Word on the course website. Each solution 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.

You are to work in groups of three or four people and should submit a single set of solutions for all problems parts designated [**Group**]. You should turn in a separate, individual solution to any problems designated [**Individual**].

We may distribute our favorite solution to each problem as the “official” solution. 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. Term Project Ideas [**Individual**]

For the 6.857 project due at end of term, you have a great degree of freedom in your choice of topic. In general, you must pick an interesting topic related to network and computer security, and are subject to common-sense restraints (regarding ethics, project size/difficulty, etc.). The topic itself, however, is yours to decide.

The term project does not need to be an implementation; it could be a literature survey, a theoretical study of some security or cryptographic mechanism, a vulnerability analysis of some real system (careful about ethics and legalities here!), a proposal for solving some security problem, etc.

Brainstorm ideas for a project that you, personally, would like to work on. Write up a brief (one-page) summary of between one and three ideas you might like. You may find the list of suggestions at <http://crypto.csail.mit.edu/classes/6.857/projects.html> useful. This is not binding in any way, and we will be discussing the project at length later; this assignment is just to get you started thinking.

The answers to this problem will be posted publicly on the class website. You will be required to work in a group of three or four for the term project. You may work in the same team as your homework group, or you may switch teams. This assignment is partially so that you may find others who are interested in working on the same things as you are; even if you are comfortable with your homework group and want to keep it for the project, this assignment will let your group get a pool of ideas to think about.

Problem 3-2. Adding One-Time Pads [**Group**]

Peter Padadder knows from 6.857 that he shouldn't ever re-use a one-time pad. He has eight messages M_1, M_2, \dots, M_8 to encrypt for his friend Mary. Each message contains the title of a sci-fi movie recommendation. The titles are of varying lengths; none are longer than 34 characters. Peter

encodes the upper case alphabet and the space character as integers modulo 27 as follows:

$$\begin{aligned}(\text{space}) &= 0 \\ A &= 1 \\ B &= 2 \\ &\vdots \\ Z &= 26\end{aligned}$$

Peter thus wants to create eight pads P_1, P_2, \dots, P_8 so that he can encrypt his messages for Mary. Peter then encrypts each message M_i 's character j as $C_{ij} = M_{ij} + P_{ij} \pmod{27}$. Note that ciphertext C_i is as long as the plaintext; if M_i is shorter than P_i then the extra pad characters are ignored.

Peter and Mary try to be clever. They create two random strings (“proto-pads”) A and B of length 34 characters each, where each character has been chosen randomly modulo 27. They then define their eight pads as:

$$\begin{aligned}P_1 &= A \\ P_2 &= B \\ P_3 &= A + B \\ P_4 &= A - B \\ P_5 &= -A \\ P_6 &= -B \\ P_7 &= -A - B \\ P_8 &= -A + B\end{aligned}$$

Peter figures this is OK since no pad is repeated. Peter encrypts his eight movie recommendations using these pads, and sends them to Mary. His encrypted messages (in arbitrary order) consist of:

```
‘COFBMEGCOFBAVGWVXXOURVIYE’
‘CESYIPCUNMAKFZRBKUGNM’
‘HMFBDKVPGUMGU HHQC’
‘X VM ONSHNRYFBFPNRUGZQQOQUPL’
‘ZIWAMFRYCCIDCPQV’
‘M WSPN GUJYR FIWSJZQYPADETTACFRA’
‘BLEIJGLAISASZZRXBLALXJHASNS’
‘XGKXLCUSXQYIUUQCCPPXHAAT GSMFMELLP’
```

These eight ciphertexts are available on the 6.857 web site at:

<http://crypto.csail.mit.edu/classes/6.857/ciphers.html>.

Show that Eve, the eavesdropper, can reconstruct these movie recommendations. Explain your work, and explain why Peter’s reasoning was faulty.

Problem 3-3. Ballot Anonymizer [Group]

Suppose an electronic polling station is going to store ballots B_0, \dots, B_{t-1} in an array of size p , where p is prime and $t < p$. Although the ballots do not have any identifying information, if they are stored sequentially, someone with access to the final ballot array could associate the ballot stored at location i with the i th voter.

Consider a scheme that places ballot B_i in array index $ai + b \pmod p$, where $a, b \in \mathbb{Z}_p^*$ and $1 \leq a \leq \frac{p-1}{2}$. This scheme is intended to jumble the ballot ordering so that someone cannot associate a ballot's storage location with a particular voter.

- (a) Explain why the ballot positions are distinct.
- (b) Suppose just three voters have cast their vote and their ballots are stored under this scheme. Explain how you can determine the order that the ballots were cast, based on where they are stored.
- (c) Suppose the adversary sees that the ballots A, B, C, D, E have been placed in memory positions $0, 1, \dots, 6$ as follows:

0	1	2	3	4	5	6
A	-	B	C	D	E	-

Can an adversary who sees this pattern of ballots in the memory array determine the order of voting? If so, how? What was the order of voting?

- (d) Show how to determine the location of the $(\frac{t-1}{2})$ th vote cast, given the final location of t ballots. You may assume that t is odd. (Hint: Consider adding up the indices of the filled memory slots.)
- (e) For some values of t , there are multiple a and b values which could lead to a given final placement of ballots. However, in other cases a final placement of ballots will uniquely specify a and b . Find out which values of t *unambiguously* define a and b values. Show how to find a and b in those cases.

You don't need to perform a brute force search. Given a and $(\frac{t-1}{2}) * a + b$, can numerically solve for b modulo p . (Hint: Consider squares of indices of filled memory slots.)

Problem 3-4. TSA Database [Group]

The Tasmanian Security Administration wants to keep a database of all people on their “no-fly” list. They wish to keep the list itself secret from casual inquiries, but want airline security personnel to be able to query the list to find out if a particular individual is on it.

However, they are concerned about an attacker spoofing the database to the security personnel. Thus, they want to publish some information in a publicly available manner (e.g. on their website, on distributed fliers, etc.) so that they can then prove (based on the public information) whether or not a name is indeed in the database. They decide to use hashing to accomplish this.

- (a) Suppose they decide that, for each person on the list, they will publish $h(ID)$, where h is some hash function and ID is some unique identifier for that person (for example, their name and date of birth). What properties must h have in order for the system to prevent casual browsing and not implicate the wrong people?
- (b) Why does this system not provide adequate secrecy of the list?

- (c) In addition to being insecure, the TSA decides that publishing a large number of hash values is inefficient from a practical perspective, and would prefer to publish just one hash value. Their next attempt is to simply publish a hash of the entire list of names.

Explain what is wrong with this system.

- (d) In order to come up with a viable solution, the TSA hires Alyssa P. Hacker, an MIT graduate, as a consultant. Alyssa quickly realizes that any solution should, on each query, have the database reveal *some* information to the airport security personnel (otherwise anyone could determine the contents of the database for themselves), but should not reveal any *secret* information beyond what was specifically queried (to protect privacy and national security). Alyssa sketches a solution in which each ID is concatenated with a long random value, then hashed using a suitably strong hash function. The TSA then publishes a hash of the concatenation of those hash values. Upon each (properly authenticated) query, the airport security employee receives back the random value associated with the queried ID, as well as the list of hashes. The computer at the airport then computes two hashes: one to verify that the ID and randomness indeed hash to a value in the list, and one to verify that the entire list hashes to the public value.

Argue that this scheme protects the secrecy of the list and does not implicate anyone not on the list.

- (e) What functional requirement is *not* met by the scheme in part (d)?
- (f) In addition to the problem in part (e), the TSA has N people on the no-fly list (N being a rather large number), and does not have the bandwidth to send N hash values each time a query is made of the database. Undaunted, Alyssa realizes that while bandwidth is expensive, computation is cheap. She comes up with a new design for the database in which each ID is hashed (without additional randomness), and then the N hashed values form the leaves of a binary tree. (You may assume that N is an exact power of 2.)

Explain how Alyssa's idea works. Include the number of values the database needs to send per query (and which ones), the number of hashes the querying computer needs to perform in order to verify that an ID is in the database, and what (single) value is published initially.

- (g) The TSA receives a request from Ken Teddedy that if an ID is *not* on the no-fly list, the database should unambiguously prove this fact to the airline security employee so that there are no misunderstandings. Alyssa points out that, if the database she proposed is modified slightly, one can show that an ID is *not* in the list. How is this done, and how many hash values does the database need to send in this case?