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

Handout 4
March 12, 2018
Due: March 26, 2018

Problem Set 3

This problem set is due on *Monday, March 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.* 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 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 3-1. Finite Fields and Elliptic Curves

In this problem, we will explore computations on finite fields and elliptic curves. The lecture notes on elliptic curves define the operations permitted on elliptic curves. You can use any programming language environment to perform your computations. Specifically you may find SageMath useful (`http://www.sagemath.org`).

(a) Finite Field Computations. Let $\mathrm{GF}(2^8)$ be defined by the polynomial $m(t) = t^8 + t^4 + t^3 + t + 1$ which is irreducible over $\mathrm{GF}(2)$. Note that, $g = t^4 + t + 1$ is a generator of the multiplicative group of $\mathrm{GF}(2^8)$.

1. Find the multiplicative inverse of $a \in \mathrm{GF}(2^8)$ where defined as $a = t^7 + t^3 + t$.
2. Find the discrete log of $t^5 + t + 1$ w.r.t g.
3. Is the element $a = t^3 + t + 1$ a cube? That is, does there exist $b \in \mathrm{GF}(2^8)$ such that $b^3 = a$?

(b) Elliptic Curves. Consider the elliptic curve defined over $\mathrm{GF}(p)$ where $p = 2017$ by the equation: $y^2 = x^3 + 5x + 3 \bmod 2017$.

1. Let $P = (60, 22)$ and $Q = (13, 128)$ be points on the curve. Compute $P + Q$.
2. Consider the elliptic curve group defined by $G = (292, 374)$. Compute $100G$.

(c) Diffie-Hellman Key Exchange. Using the same curve, and the point G, Alice and Bob would like to share a key using the (Elliptic Curve) Diffie-Hellman key exchange protocol. Alice picks $s_A = 21$ and Bob picks $s_B = 35$. What messages do they send each other, and what is their shared secret? Show your work and be sure to verify that Alice computes the same secret as Bob.

Problem 3-2. Message Authentication Codes

Alice wants to send to Bob an encrypted and authenticated message M. Alice and Bob share a random (long) secret key K. In what follows, we use the standard notion of a secure MAC (i.e., security against adaptive chosen message attacks).

(a) What can go wrong if they use the same key K both for encryption and for authentication? Give an example of a CPA secure encryption scheme and a secure MAC scheme, such that if Alice and Bob use the same key for both then security breaks.

(b) Alice and Bob decide to use CMAC as their MAC scheme. Recall that CMAC applies the CBC mode of operation to the message, while using a fresh key for the last block, and outputs only the last ciphertext block. Moreover, CMAC uses $IV = 0$.

1. What goes wrong if Alice and Bob use the same key for all blocks (including the last block)?
2. What goes wrong if Alice and Bob output all the blocks, not only the last block?
3. Suppose that instead Alice and Bob, decide choose a random IV, and include IV as part of the MAC for verification. Is this secure?

(c) Alice and Bob have a secure MAC that can authenticate only messages consisting of 128 bits. Suppose they wish to authenticate a message consisting of 256 bits. How can they use the underlying MAC? We have proposed two options. Show why each of these options is or is not secure.

1. One option is to partition the message M into two parts $M = (M_1, M_2)$ where each M_i is of length 128, and MAC each part separately using the following scheme. Alice and Bob select and share a random, secret key $k \in \{0,1\}^{128}$. To authenticate a message $M_1 \parallel M_2$ with $|m_1| = |m_2| = 128$, they compute the MAC: $MAC_k(M_1) \parallel MAC_k(MAC_k(M_2))$.
2. Another option is to use a hash function function $H : \{0,1\}^{256} \to \{0,1\}^{128}$ and output the MAC of $H(M)$—that is, $MAC_k(H(M))$. Is this secure if H is a one-way function? Explain your answer.

Problem 3-3. Blakley's Secret Sharing Scheme

In class, we saw Shamir's secret sharing scheme, in which a dealer that knows a secret s can generate a set of n shares S such that any subset of the shares $T \subseteq S$, where $|T| \geq t$, can be used to recover the secret while the secret is hidden given any $T' \subset S$ shares where $|T'| < t$. This type of secret sharing scheme is called a t-out-of-n *threshold* secret sharing scheme.

In this problem, we will explore another threshold secret sharing scheme called Blakley's secret sharing scheme. We will describe the t-out-of-n threshold secret sharing scheme in a simplified setting. Like Shamir's scheme, Blakley's scheme is also defined over a finite field: $\mathbb{Z}_p$ for a prime $p > n$. To share the secret $s \in \mathbb{Z}_p$, the dealer first samples a point $\vec{s} = (s, s_2, s_3, \ldots s_t) \in \mathbb{Z}_p^t$ where $s_2, s_3, \ldots s_t \leftarrow \mathbb{Z}_p$ are picked at random.

To generate shares for the n parties, the dealer randomly picks n distinct non-zero field elements $\{a_i\}_{i \in [n]} \in \mathbb{Z}_p$. Each a_i describes a hyperplane H_i as follows: H_i is the hyperplane $a_i x_1 + a_i^2 x_2 + \ldots a_i^t x_t = b_i$ such that the point $\vec{s}$ lies in the hyperplane. That is, $a_i s + a_i^2 s_2 + \ldots a_i^t s_t = b_i$. Then player i is given the share (a_i, b_i).

(a) Give the recovery algorithm for Blakley's secret sharing scheme (i.e. give an algorithm **Recover**(T) that takes as input a subset T of t or more shares and outputs the underlying secret s).

(b) Prove that Blakley's scheme is information-theoretically secure (i.e. given any $t-1$ shares any secret s, no adversary can learn any information about the secret).

(c) We have set up a server at `http://6857blakley.csail.mit.edu` which when queried returns an id along with three shares of a random secret under a 3-out-of-3 Blakley's scheme. For this part of the problem, query the server, reconstruct the secret shared and in your answer include the id, shares and the recovered secret.

We have provided our server implementation in the files `server.py` and `ffield.py`. The server is using the finite field $\mathbb{Z}_p$ where $p = 11953696440786470837$. Feel free to play with it on your local machine.