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

Handout 4
March 24, 2021
**Due:** April 5, 2021

# Problem Set 3

This problem set is due on *Monday, April 5, 2021* 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. *Your solutions to all problems should be written up in a single pdf.* Have **one and only one group member** submit the finished pdf containing the problem writeups. Please title the PDF with the Kerberos of your group members as well as the problem set number. (i.e. *kerberos1_kerberos2_kerberos3_pset2.pdf*).

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-staff@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!* 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.

## Problem 3-1. Verifiable Secret Sharing

In lecture, we have looked at Shamir's secret sharing scheme, a method that allows a group of $n$ individuals to hide a secret by assigning shares to each individual such that no less than $t+1$ shares can retrieve the secret. Although effective, Shamir's scheme does not offer any guarantee regarding the integrity of the shares that are distributed. For example, the dealer (the entity that distributes the shares) may be malicious and could distribute invalid shares to the participants.

The notion of Verifiable Secret Sharing (VSS) exists to address this potential problem. One VSS variant, Feldman's secret sharing scheme, builds on Shamir's scheme and offers share integrity by making the dealer commit to the polynomial concealing the secret. Feldman's scheme is as follows (pay attention to when we use $p$ and when we use $q$):

1. First, a cyclic group $G$ with prime order $p$ for which the discrete log assumption holds is publicly chosen, along with a generator of $G$. For this problem, assume that $p$ is a Sophie-Germain prime and that $q = 2p + 1$. Also assume that $G = \mathbb{Q}_q$, which has order $p$. Let $g$ be the chosen generator for this group.
2. The dealer is given a secret $s$ and computes a random polynomial $P$ of degree $t$ with coefficients in $\mathbb{Z}_p$ without revealing it. Let the computed polynomial be $P(x) = s + a_1x + a_2x^2 + ... + a_tx^t$.
3. For $j \in \{1, 2, ..., n\}$, each shareholder picks a unique value $x_j$ to generate their share. The dealer gives the share $v_j = P(x_j) \pmod p$ to shareholder $j$.
4. The dealer also computes commitments to the coefficients of the polynomial $P$ in modulo $q$. These commitments are $c_0 = g^s \pmod q$ and $c_i = g^{a_i} \pmod q$ for $i \in \{1, 2, ..., t\}$. The dealer also gives all shareholders its commitments $c_0, c_1, ..., c_t$ to the coefficients of $P$.

**Note:** Step 1 defines $p$ and $q$. We use only $p$ in steps 2 and 3, and we use only $q$ in step 4.

**(a)** Suppose a participant is given share $v_k = P(x_k) \pmod p$ where $k$ is chosen uniformly at random from $\{1, 2, ..., n\}$. Describe how the participant can use the commitments $c_0, c_1, ..., c_t$ to determine whether or not $v_k$ is a valid share. Explain why your strategy works. (Hint: How can the participant compute $g^{P(x_k)}$ using the values they have access to?)

**(b)** Let's test out the strategy you described in part (a). The 6.857 course staff has run Feldman's scheme to safeguard an important secret. Our dealer has generated 50 shares total for us to distribute, and no less than 25 shares should be able to retrieve our secret. On the course website, you can find two files, one containing the 50 shares and the other containing the dealer's commitments to the polynomial coefficients. In `shares.txt`, each line corresponds to a share. More specifically, each line contains a comma-separated pair of which the first value is the $x_k$ chosen by the shareholder and the second value is the corresponding share returned by the dealer. In `commitments.txt`, the first line contains the integer modulo $q$, the second line contains the value of the selected generator $g$, and each subsequent line contains commitments to the coefficients of the dealer's polynomial in order of increasing degree. Our dealer was corrupted and has given 5 invalid shares. Determine which of the 5 shares given by the dealer are invalid. In your write-up, be sure to include both your code and the shares (or indices of the shares) that you find to be invalid. Feel free to use any language for your implementation.

**(c)** Suppose you have a malicious shareholder who is **NOT** computationally bounded. How can this shareholder bypass the secret sharing scheme and access the secret without working with any of the other shareholders?

## Problem 3-2. Key Exchange

In class, we learned about Diffie-Hellman key exchange, and we showed that it is secure against passive attacks under the Decisional Diffie-Hellman assumption. In our context, passive attackers are adversaries that only listen to the communication (i.e., can see any and all messages between Alice and Bob), but do not tamper with it. In particular, an adversary can not modify, delete nor insert messages.

**(a)** Construct an active adversary, that may tamper with the communication, which breaks the security of the Diffie-Hellman key exchange. For example, demonstrate an active adversarial behavior that will allow the adversary to learn the secret key (i.e., $g^{xy}$) computed by Alice (or Bob).

**(b)** Recall that the Diffie-Hellman key exchange protocol uses a group $G$ with a generator $g$. Is it secure if we choose a random $k$-bit prime $p$ (where $k$ is a security parameter), set $G = Q_p$ and let $g$ be a random element in $Q_p$? Explain why or give a counterexample.

## Problem 3-3. Authenticating El-Gamal

In class, we learned about El-Gamal encryption. As a reminder, El-Gamal operates as follows, with algorithms $(\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$. Suppose Alice wants to be able to receive a confidential message. Alice runs `Gen` which chooses a group $G$ of prime order $p$ and generator $g$, as well as $g^s$ for a random $s \leftarrow \mathbb{Z}_p$. Alice can then publish the tuple $(G, p, g, g^s)$ as her public key. Then any user Bob can use `Enc` to encrypt a message $m \in G$ to Alice given her public key $(G, p, g, g^s)$, by generating a ciphertext $\texttt{CT} = (v, c)$, where $v = g^r$ for a random $r \leftarrow \mathbb{Z}_p$ and $c = m(g^s)^r = mg^{sr}$. Finally, to decrypt the ciphertext $(v, c)$, Alice uses `Dec` and her secret key $s$ to raise $v$ to the power $s$, $v^s = g^{sr}$, and then multiply $c$ by the inverse of the result: $c(v^s)^{-1} = mg^{sr}(g^{sr})^{-1} = m$.

**(a)** Is El-Gamal CCA secure? Explain or give a counterexample.

**(b)** Suppose we want to authenticate an El-Gamal ciphertext. Consider the following attempt to modify El-Gamal using a MAC, as follows:

`Gen` stays the same, but `Enc` is modified to generate, in addition to $\texttt{CT} = (g^r, mg^{sr})$, also $t = \texttt{MAC}(g^{sr}, \texttt{CT})$. Namely, it uses the "one-time pad" $g^{sr}$ used to pad the message $m$ also to MAC the ciphertext `CT`. It outputs $(\texttt{CT}, t)$. The decryption algorithm, in addition to decrypting `CT` by using the secret key $s$ to compute the "one-time pad" $g^{sr}$ and using it to unmask the message $m$, also checks the validity of the tag $t$ using $g^{sr}$ as the secret key, and outputs $m$ only if the tag is valid.

Is this modified El-Gamal scheme CCA secure? Explain why or give a counterexample.