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

Handout 5
March 30, 2020
Due: April 13, 2020

Problem Set 4

This problem set is due on *Monday, April 13, 2020* 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 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-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 4-1. 6857coin Rumor has it that a new cryptocurrency has sprung up at MIT!

In 6857coin, a proof-of-work called FiveSwap is used, where a block is considered to be valid if the following procedure returns true (more details and definitions on the 6857coin website (`http://6857coin.csail.mit.edu/`):

1. Generate array X where $X[0]$ is the hash of the block and for $0 < i < L$, $X[i] = FiveSwap(x[i-1])$.
2. Generate array Y where $Y[i] = X[i] + X[L-i+1]$ for $0 \leq i < \frac{L}{2}$
3. Count the total number of 1 bits in the binary representation of the values in Y.
4. Return true if the number of 1 bits is greater than the difficulty.

The FiveSwap POW is intended to emphasize memory usage over computation power.

(a) Argue that the FiveSwap function is one-to-one for w-bit words.

(b) Implement this POW, and extend the class coin chain if you can. Set the content of the block to be the names of all members of your team (comma-delimited). Template code to add blocks is available on the class website. Include your code in your PDF submission.

(c) Argue that this POW is not necessarily "memory hard" – it be computed with substantially less than L w-bit words of memory.

(d) What other pros and cons does FiveSwap have compared to other POW schemes?

Problem 4-2. Digital Signatures and Hash-and-Sign

The security definition of a digital signature scheme takes into account both the **adversary's goal** and the **power of the adversary**.

From strongest to weakest, some examples of the adversary goals are:

- **Total break/ Universal forgery** - the signature scheme can be completely broken and the adversary is now able to forge signatures for any arbitrary message.
- **Random forgery** - the adversary is able to generate valid signatures for random message.
- **Existential forgery** - the adversary is able to find at least one message for which they are able to provide a valid signature without querying for it.

On the other hand, the adversary's powers from weakest to strongest can be:

- **Key-only attack** - the adversary only has the public-key of the signature scheme.
- **Random-message attack** - the adversary has oracle access to a message signer which returns a random message and its corresponding signature.
- **Chosen-message attack** - at the start of the attack, the adversary picks a number of messages and receives valid signatures for these messages.
- **Adaptive chosen-message attack** - throughout the attack, the adversary has oracle access to a message signer that can sign any message chosen by the adversary.

We say a scheme is **secure** if it satisfies the strongest definition of security, i.e. **existential unforgeability** under **adaptive chosen message attack**.

(a) It was shown in class that the textbook RSA signature scheme is not **secure**. However, hash-and-sign RSA is **secure** in the random oracle model (ROM).

Consider an arbitrary signature scheme $(\text{Gen}, \text{Sign}, \text{Ver})$. What is the minimal security definition (as above defined using the adversary's goal and allowed powers) does this signature scheme need to have so that the hash-and-sign version of is **secure** in the ROM? Explain your answer.

(b) In addition to improving the security of signature schemes, hash-and-sign is also used to improve efficiency.

Let $(\text{Gen}, \text{Sign}, \text{Ver})$ be a **secure** signature scheme. What property/properties of hash functions are necessary for the hash-and-sign variation to still be **secure**? Explain your answer.

Problem 4-3. Zero Knowledge Proofs and Commitments

In class, we saw a zero knowledge proof for graph 3-colorability. Specifically, in this proof, the Prover applies a random permutation to the colors of the nodes, and sends a commitment to all these permuted colors; the Verifier sends a random edge; and the prover opens the commitments to the colors corresponding to the nodes of that edge. The commitment scheme is assumed to be hiding, that is, the Verifier learns nothing about the values committed to (unless the Prover opens the commitment), and binding, that is, the Prover cannot open a commitment to two different values.

(a) Explain what goes wrong in the following two cases, and how zero-knowledge and soundness are affected.

1. The commitment scheme is binding but not hiding.
2. The commitment scheme is hiding but not binding.

(b) We examined two different commitment schemes. One was perfectly binding and computationally hiding, while the other was perfectly hiding and computationally binding. The second, known as Pedersen's commitment scheme, is "keyed", that is, the commitment function is associated with a key (g, h), where g, h are elements of group G of prime order p. To commit to a message $m \in \mathbb{Z}_p$, choose a random element in $r \in \mathbb{Z}_p$ and send $c = g^m \cdot h^r$.

1. Is the commitment scheme secure if h is random but g is a *fixed* generator?
2. Suppose g is a random generator. Is the commitment scheme secure if h is not random, but a *fixed* power of g ($h = g^s$)?

3. Suppose that instead of using a prime order group G, we use $G = \mathbb{Z}_p^*$, and think of the message m and the randomness r as elements in $\mathbb{Z}_{|G|}$. Let g be a random generator and h a random element. Is this commitment scheme secure?