Handout 5
April 2, 2018
Due: April 23, 2018

Problem Set 4

This problem set is due on *Monday, April 23, 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 4-1. El-Gamal Encryption and Pedersen Commitments

(a) Alice and Bob decide the Pedersen commitment scheme is too slow for them (repeated exponentiation is very computationally expensive). Help Alice and Bob by *describing* an alternative commitment scheme that uses a hash function H such that the scheme is (a) hiding and binding in the random oracle model and (b) non-malleable in the random oracle model.

(The new scheme should not use any exponentiations.)

(b) Let G be a group of order t. Let a be some element of G. Let $k \geq 0$ be an integer such that $\gcd(k, t) = 1$. Show that $\text{order}(a) = \text{order}(a^k)$.

(c) Alice and Bob have heard about the El-Gamal encryption scheme and decide to implement it in the following way: they choose a large, safe prime p (i.e., $(p-1)/2$ is also a prime), choose a random generator $g \in \mathbb{Z}_p^*$, and set $PK = (g, y = g^x)$ and $SK = x$. They encrypt any message $m \in \mathbb{Z}_p^*$ by choosing a random k and outputting $(g^k, y^k \cdot m)$. Is this secure? Explain.

(d) Alice and Bob decide to use the same distribution (g, y) as above for producing Pedersen Commitments. Namely, Alice, upon receiving parameters (g, y) as above, commits to a $m \in \{0, 1, \dots p-2\}$ by choosing random r and sending $g^m \cdot y^r$ to Bob. Is this scheme binding? Is it hiding? Explain.

Problem 4-2. Digital Signatures

(a) Alice is worried about leakage when using Full Domain Hash (FDH) RSA signatures, and therefore decides to use $n = pqr$ as her public key, where p, q, r are random 1024-bit primes (instead of $n = pq$). Let H be a hash function. The public key $PK = (n, e)$ and the secret key is[1] $SK = (n, d)$ where $de \equiv 1 \pmod{\phi(n)}$. The Signing and Verification algorithms are as follows:

- $\text{Sign}((SK, H), m) = (H(m))^d \bmod n$.

[1]The previous version of the pset had $SK = n, p, q, e$.

- Verify$((PK, H), m, \sigma) = 1$ if and only if $\sigma^e = H(m) \bmod n$.

Is the new scheme as secure as the original one where $n = pq$ where p, q are random 1024-bit primes? Does it remain secure if one of the prime factors is leaked? Explain.

(b) You do not have to answer anything for this part. In the previous version it was the set-up for the next part, which has now been moved to the next part.

(c) We saw in class that any trapdoor permutation can be used to construct a digital signature. However, the only trapdoor permutation candidate known is the RSA permutation.

Let p, q be random 1024-bit primes such that $p, q \equiv 3 \pmod 4$, and let $n = pq$. Consider the function $f : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$, defined by $f(x) = x^2 \bmod n$.

Show that f is a trapdoor function. Namely, show how one can invert f (efficiently) given the prime factors p, q, and argue that f is hard-to-invert given only n (without knowing p, q), assuming the hardness of factoring.

(For the latter part, show that one can factor n in to primes p, q, given black-box access to an algorithm that given $y \in \mathbb{Z}_n^*$ outputs x such that $x^2 \equiv y \bmod n$ (if such a square root x exists).)

(d) Alice would like to use this trapdoor function f to construct a digital signature scheme, as suggested in the seminal paper of Diffie and Hellman. Namely, to compute a signature of m simply compute $f^{-1}(H(m))$ using the trapdoor. However, our function f is not a permutation, and $H(m)$ may not have an inverse (i.e., a square root). Suggest a way to modify this scheme to overcome this problem, so that every message can be signed (given the trapdoor p, q).

Problem 4-3. 6857coin

Rumor has it that a new cryptocurrency has sprung up at MIT!

In 6857coin, a proof-of-work called AESHAM2 is used and it expects 3 nonces. It first computes two AES keys from the first nonce. Then it expects the cross sums of the AES ciphertexts on the next two nonces to have small hamming distance. For more details, please refer to the website.

Note: this problem is somewhat of an experiment for us, and we reserve the right to tweak it with reasonable warning as events unfold.

(a) To get started, visit `http://6857coin.csail.mit.edu/` and read the API for 6857coin. Then, look at the provided miner.py template and make the required modifications to begin mining. You will receive full credit for part (a) after successfully mining a block that appends to any tree rooted at the genesis block. To receive credit for your team, include your team members' usernames separated by commas in the block contents.

(b) Now see where you can optimize your miner even further. The slower your miner is in comparison to other miners, the longer it will take to add to the main (longest) chain. You will receive full credit for part (b) if you ever append to the main chain and a description of your strategy for mining a block on the main chain. Remember to include your team in the block contents! Also note that the earlier you start, the slower your competition will be! Feel free to get creative by using different languages or hardware.