CS 355: Topics in Cryptography **Spring 2020**

Problem Set 1

Due: April 20, 2020 at 11:59pm PT (submit via Gradescope)

Instructions: You **must** typeset your solution in LaTeX using the provided template:

https://crypto.stanford.edu/cs355/20sp/homework.tex

Submission Instructions: You must submit your problem set via Gradescope. Please use course code **975GG7** to sign up. Note that Gradescope requires that the solution to each problem starts on a **new page**.

Bugs: We make mistakes! If it looks like there might be a mistake in the statement of a problem, please ask a clarifying question on Piazza.

Problem 1: Conceptual Questions [8 points]. For each of the following statements, say whether it is TRUE or FALSE. Write *at most one sentence* to justify your answer.

(a) Which of the following are true in a world where $\mathsf{P} = \mathsf{NP}$.

i Secure PRFs exist in the standard model.

ii Secure PRFs exist in the random oracle model.

iii The one-time-pad cipher is secure.

(b) If there exists a PRG with 1-bit stretch, there exists a PRG with n^{800}-bit stretch (where n is the length of the PRG seed).

(c) Let $P\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ be a pseudorandom permutation. Then:

i $f_{k=0}(x) := P(0,x)$ is (always) a one way function.

ii $f_{k=0}(x) := P(0,x)$ is (always) a one way permutation.

iii $f_{x=0}(k) := P(k,0)$ is (always) a one way function.

iv $f_{x=0}(k) := P(k,0)$ is (always) a one way permutation.

Problem 2: Key Leakage in PRFs [5 points]. Let F be a secure PRF defined over $(\mathcal{K},\mathcal{X},\mathcal{Y})$, where $\mathcal{K} = \mathcal{X} = \mathcal{Y} = \{0,1\}^n$. Let $\mathcal{K}_1 = \{0,1\}^{n+1}$. Construct a new PRF F_1, defined over $(\mathcal{K}_1,\mathcal{X},\mathcal{Y})$, with the following property: the PRF F_1 is secure; however, if the adversary learns the last bit of the key then the PRF is no longer secure. This shows that leaking even a *single* bit of the secret key can completely destroy the PRF security property.
[**Hint:** Try changing the value of F at a single point.]

Problem 3: A weak form of the Goldreich-Levin Theorem [10 points]. We say that $b\colon \{0,1\}^* \to \{0,1\}$ is a hard*ish*-core bit of a permutation f if $b(x)$ is efficiently computable given x, and for every efficient algorithm $\mathcal{A}$ it holds that

$$\Pr[\mathcal{A}(f(x)) = b(x) : x \xleftarrow{R} \{0,1\}^n] \leq 3/4 + \epsilon$$

for every positive constant ϵ.[1] We will prove that if $f\colon \{0,1\}^n \to \{0,1\}^n$ is a one-way permutation, then $b(x,r) = \langle x,r\rangle$ is a hardish-core bit of the permutation $g\colon \{0,1\}^{2n} \to \{0,1\}^{2n}$ defined as $g(x,r) = (f(x),r)$ for $x,r \in \{0,1\}^n$. Here $\langle x,r\rangle$ denotes the inner product of x and r mod 2.

For the sake of contradiction, suppose that there exists a constant $\epsilon > 0$, and an algorithm $\mathcal{A}$ that takes as input $f(x), r \in \{0,1\}^n$ and outputs a bit, such that

$$\Pr_{x,r}[\mathcal{A}(f(x),r) = \langle x,r\rangle] > 3/4 + \epsilon,$$

where the probability is taken over both x and r, each chosen uniformly from $\{0,1\}^n$. We will construct an algorithm $\mathcal{B}$ that breaks the one-wayness of f.

(a) As a warmup, suppose first that $\Pr[\mathcal{A}(f(x),r) = \langle x,r\rangle] = 1$. Show how to construct an efficient algorithm $\mathcal{B}$ that perfectly inverts f (i.e., given $f(x)$ outputs x).

(b) Next, we say that $x \in \{0,1\}^n$ is *good* for $\mathcal{A}$ if $\Pr_r[\mathcal{A}(f(x),r) = \langle x,r\rangle] > 3/4 + \epsilon'$ for some positive constant ϵ', where the probability is taken *only* over $r \xleftarrow{R} \{0,1\}^n$. Construct an efficient algorithm $\mathcal{B}$ that takes as input $f(x)$ for a good $x \in \{0,1\}^n$ and outputs x with probability at least $1/2$, by calling $\mathcal{A}$ at most $O(n \cdot \log n)$ times.

(c) Show that x chosen uniformly from $\{0,1\}^n$ is *good* with some constant probability. Conclude that algorithm $\mathcal{B}$ breaks the one-wayness of f.

Problem 4: Vector Commitments [10 points]. Let $\mathbb{G}$ be a group of prime order q in which the discrete logarithm problem is hard. Let g and h be generators of $\mathbb{G}$. As we saw in class, the Pedersen commitment scheme commits to a message $m \in \mathbb{Z}_q$ using randomness $r \in \mathbb{Z}_q$ as $\mathsf{Commit}(m;r) := g^m h^r \in \mathbb{G}$. Moreover, we saw that Pedersen commitments are *additively homomorphic*, meaning that given commitments to m_1 and m_2, one can compute a commitment to $m_1 + m_2$. The "public parameters" associated with the Pedersen commitment scheme are the description of the prime-order group $\mathbb{G}$ and the group elements g and h.

(a) Use $\mathbb{G}$ to construct an additively homomorphic commitment scheme $\mathsf{Commit}_n(m_1,\ldots,m_n;r)$ that commits to a length-n vector of messages $(m_1,\ldots,m_n) \in \mathbb{Z}_q$ using randomness $r \in \mathbb{Z}_q$. The output of the commitment should be short – only a single group element. You should specify both the public parameters of your scheme (which may be different from that of the basic Pedersen commitment scheme) as well as the description of the Commit_n function.

(b) Prove that your commitment scheme is perfectly hiding and computationally binding (assuming hardness of discrete log in $\mathbb{G}$).

(c) Show that if you are given a hash function $H\colon \mathbb{Z}_q \to \mathbb{G}$ (modeled as a random oracle), the public parameters for your construction from Part (a) only needs to consist of the description of the group

[1] An actual hard-core bit cannot be guessed with probability noticeably better than $1/2$.

$\mathbb{G}$ and the description of H. Argue *informally* why your construction is secure. You do *not* need to provide a formal proof. This problem shows that getting rid of public parameters is another reason why random oracles are useful in practice!

(d) **Extra Credit [5 points].** Prove formally that your construction from Part (c) is secure in the random oracle model.

Problem 5: Feedback [0 points]. Please answer the following questions to help us design future problem sets. You are not required to answer these questions, and if you would prefer to answer anonymously, please use this form. However, we do encourage you to provide us feedback on how to improve the course experience.

(a) Roughly how long did you spend on this problem set?

(b) What was your favorite problem on this problem set?

(c) What was your least favorite problem on this problem set?

(d) Any other feedback for this problem set?