CS 388H: Cryptography **Fall 2021**

Homework 3: Symmetric and Public-Key Cryptography

Due: October 28, 2021 at 11:59pm (Submit on Gradescope) **Instructor:** David Wu

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

https://www.cs.utexas.edu/~dwu4/courses/fa21/static/homework.tex

You must submit your problem set via Gradescope (accessible through Canvas).

Collaboration Policy. You may discuss your general *high-level* strategy with other students, but you may not share any written documents or code. You should not search online for solutions to these problems. If you do consult external sources, you must cite them in your submission. You must include the names of all of your collaborators with your submission. Refer to the official course policies for the full details.

Problem 1: Goldreich-Levin and List Decoding [28 points]. Fix any $x \in \mathbb{Z}_2^n$ and let $f_x : \mathbb{Z}_2^n \to \mathbb{Z}_2$ be a function where

$$\Pr[r \xleftarrow{R} \mathbb{Z}_2^n : f_x(r) = \langle x, r \rangle] \geq p. \tag{1}$$

In lecture, we showed that when $p = 3/4 + \varepsilon$ for any constant $\varepsilon > 0$, there exists an efficient algorithm $\mathcal{A}^{f_x}$ that makes $O(n \log n)$ queries to f_x and outputs x with probability at least $1/2$. We often refer to $\mathcal{A}$ as a "unique decoder." In this problem, we consider the setting where $p = 1/2 + \varepsilon$ for a constant $\varepsilon > 0$.

(a) Show that when $p = 1/2 + \varepsilon$, the point x may *not* be uniquely determined. In particular, show that there exists $x \neq y \in \mathbb{Z}_2^n$ and a single function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ such that

$$\Pr[r \xleftarrow{R} \mathbb{Z}_2^n : f(r) = \langle x, r \rangle] \geq p$$
$$\Pr[r \xleftarrow{R} \mathbb{Z}_2^n : f(r) = \langle y, r \rangle] \geq p,$$

for some $p \geq 1/2 + \varepsilon$ where $\varepsilon > 0$. **Hint:** Consider $x, y \in \mathbb{Z}_2^n$ that differ on a single index.

This shows that given a function f_x satisfying Eq. (1) with $p > 1/2 + \varepsilon$, any algorithm $\mathcal{A}$ that is constrained to outputting a *single* point cannot recover x with probability better than $1/2$. Instead, we relax our requirement to allow $\mathcal{A}$ to output a list $L \subset \mathbb{Z}_2^n$ of *possible* x's, and we say that $\mathcal{A}$ succeeds if $x \in L$ and $|L| = \text{poly}(n)$. We refer to $\mathcal{A}$ as a "list decoder."

(b) Suppose that in addition to f_x, you have access to an oracle $\mathcal{O}_x$. When invoked, $\mathcal{O}_x$ samples $r \xleftarrow{R} \mathbb{Z}_2^n$ and outputs $(r, \langle x, r \rangle)$. Oracle $\mathcal{O}_x$ outputs a *fresh* sample on every invocation. Show how to construct an efficient algorithm $\mathcal{B}^{f_x, \mathcal{O}_x}$ where for all $y \in \mathbb{Z}_2^n$,

$$\Pr[\mathcal{B}^{f_x, \mathcal{O}_x}(y) = \langle x, y \rangle] \geq 1 - 1/(10n).$$

Algorithm $\mathcal{B}^{f_x, \mathcal{O}_x}$ can make at most $k = O(\log n)$ queries to $\mathcal{O}_x$ and f_x. Note that x is uniquely defined by the oracle $\mathcal{O}_x$.

(c) Show how to use algorithm $\mathcal{B}$ to construct an efficient list decoder $\mathcal{A}^{f_x}$ that makes $\text{poly}(n)$ queries to f_x and outputs a list $L \subset \mathbb{Z}_2^n$ of size $\text{poly}(n)$ such that $x \in L$ with probability at least $9/10$. Note that $\mathcal{A}$ does *not* have access to the oracle $\mathcal{O}_x$. **Hint:** Let $k = c \log n$ be an upper bound on the number of queries $\mathcal{B}$ makes to $\mathcal{O}_x$ where $c > 0$ is a constant. Observe that $2^k = \text{poly}(n)$.

(d) The full Goldreich-Levin theorem gives an efficient algorithm for list decoding for any $p = 1/2 + \varepsilon$ whenever $1/\varepsilon = \text{poly}(n)$. Briefly explain what goes wrong in our above approach when $1/\varepsilon = \omega(1)$ (i.e., the distinguishing advantage is sub-constant).

Remark: While the specific approach we took in this problem does not generalize to the setting where $p = 1/2 + 1/\text{poly}(n)$, a slight variant of this basic approach does. We can then use a similar averaging argument from lecture to translate this statement about list decoding back to the setting of one-way permutations and the Goldreich-Levin hard-core bit.

Problem 2: DDH in Composite-Order Groups [12 points]. Let $k > 1$ be a constant, and let $\mathbb{G}$ be a cyclic group of order kq where $\gcd(k, q) = 1$. Let g be a generator of $\mathbb{G}$. Show that the DDH assumption does not hold in $\mathbb{G}$. (Specifically, for this setting, the DDH assumption asserts that the distributions of (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^r) where $a, b, r \xleftarrow{R} \mathbb{Z}_{kq}$) are indistinguishable to any efficient adversary. You can assume that k and q are known to the adversary. **Hint:** If $x = y \bmod kq$, then $x = y \bmod k$.

Remark: This shows that the DDH assumption does not hold in $\mathbb{Z}_p^*$ (for prime p) when $p = kq + 1$. In fact, the DDH assumption does not hold in $\mathbb{Z}_p^*$ for any prime p (there is an efficient distinguisher based on Legendre symbols). However, assumptions such as CDH or discrete log still plausibly hold over $\mathbb{Z}_p^*$.

Problem 3: Hash Functions from Discrete Log [15 points]. Let $\mathbb{G}$ be a group of prime order p with generator g. Sample $h_1, \ldots, h_n \xleftarrow{R} \mathbb{G}$ and define the hash function $H_{h_1,\ldots,h_n} \colon \mathbb{Z}_p^n \to \mathbb{G}$ as follows:

$$H_{h_1,\ldots,h_n}(x_1, \ldots, x_n) := h_1^{x_1} h_2^{x_2} \cdots h_n^{x_n} \in \mathbb{G}.$$

(a) Show that $H_{h_1,\ldots,h_n}$ is collision resistant under the discrete log assumption in $\mathbb{G}$. Specifically, in the (keyed) collision-resistant hashing security game, the adversary is first given $h_1, \ldots, h_n \xleftarrow{R} \mathbb{Z}_p$ and it succeeds if it outputs $(x_1, \ldots, x_n) \neq (x'_1, \ldots, x'_n)$ such that $H_{h_1,\ldots,h_n}(x_1, \ldots, x_n) = H_{h_1,\ldots,h_n}(x'_1, \ldots, x'_n)$. In the discrete log security game, the adversary is given $h \xleftarrow{R} \mathbb{G}$, and it wins if it outputs $x \in \mathbb{Z}_p$ such that $h = g^x$. **Hint:** Consider a reduction algorithm that starts by guessing the index $i^* \in [n]$ (uniformly at random) where $x_{i^*} \neq x'_{i^*}$. Show that your reduction algorithm succeeds whenever the guess is correct. Remember to compute the advantage of your reduction algorithm (for breaking the discrete log assumption).

(b) Show that the function $H_{h_1,\ldots,h_n}$ has a *trapdoor* that can be used to sample pre-images. Specifically, show that if someone knew the discrete logs of $h_1, \ldots, h_n$ (i.e., $z_i \in \mathbb{Z}_p$ where $h_i = g^{z_i}$ for each $i \in [n]$), then for any $t \in \mathbb{Z}_p$, they can find a pre-image $(x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ such that $H_{h_1,\ldots,h_n}(x_1, \ldots, x_n) = g^t$.

Problem 4: Encrypted Group Chat [20 points]. Suppose a group of n people (denoted $P_1, \ldots, P_n$) want to set up a shared key for an encrypted group chat. At the end of the group key-exchange protocol, everyone within the group should know the key, but an eavesdropper on the network should not. We will use the following variant of Diffie-Hellman over a group $\mathbb{G}$ of prime order p and generator g:

- At the beginning of the protocol, P_1 chooses $s \xleftarrow{R} \mathbb{Z}_p$. We will view P_1 as the group administrator that all of the other parties know.
- Each of the other parties P_i ($2 \le i \le n$) samples $r_i \xleftarrow{R} \mathbb{Z}_p$ and sends $x_i \leftarrow g^{r_i}$ to the group administrator P_1. The administrator P_1 replies to P_i with x_i^s.
- The group key is then defined to be $k \leftarrow H(g^s)$, where $H\colon \mathbb{G} \to \{0,1\}^\lambda$ is a hash function.

Both the group description $(\mathbb{G}, p, g)$ and the hash function H are public and known to everyone (both the protocol participants and the eavesdropper).

(a) Show that both the group administrator P_1 and each of the parties P_i ($2 \le i \le n$) are able to efficiently compute the group key.

(b) We say that the group key-exchange protocol is secure against eavesdroppers if no efficient adversary who sees the transcript of messages sent by the honest parties $P_1, \ldots, P_n$ is able to distinguish the group key k from a uniform random string over $\{0,1\}^\lambda$, except perhaps with negligible probability. If we model H as an "ideal hash function" (i.e., random oracle), it suffices to argue that the shared Diffie-Hellman secret g^s is *unguessable*: namely, for all efficient adversaries $\mathcal{A}$,

$$\Pr[\mathcal{A}(x_2, x_2^s, \ldots, x_n, x_n^s) = g^s] = \mathrm{negl}(\lambda), \tag{2}$$

where $x_i = g^{r_i}$ and $r_2, \ldots, r_n, s \xleftarrow{R} \mathbb{Z}_p$. This means that an eavesdropper who only observes the messages sent by the honest parties cannot guess g^s, and correspondingly, the shared key $H(g^s)$ is uniformly random and unknown to the adversary.

Show that under the CDH assumption in $\mathbb{G}$, the shared Diffie-Hellman secret g^s in the group key-exchange protocol above is unguessable (i.e., Eq. (2) holds for all efficient adversaries $\mathcal{A}$). As usual, you should consider the contrapositive: show that if there exists an efficient adversary $\mathcal{A}$ that can predict g^s from the above challenge tuple $(x_2, x_2^s, \ldots, x_n, x_n^s)$, then there exists an efficient algorithm $\mathcal{B}$ that breaks CDH in $\mathbb{G}$. **Hint:** Your algorithm $\mathcal{B}$ may need to invoke $\mathcal{A}$ *more than once*. Remember to compute the advantage of the adversary you construct.

Problem 5: Time Spent [3 extra credit points]. How long did you spend on this problem set? This is for calibration purposes, and the response you provide does not affect your score.

Optional Feedback. Please answer the following *optional* questions to help us design future problem sets. You do not need to answer these questions. However, we do encourage you to provide us feedback on how to improve the course experience.

(a) What was your favorite problem on this problem set? Why?

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

(c) Do you have any other feedback for this problem set?

(d) Do you have any other feedback on the course so far?