CS 388H: Cryptography **Fall 2022**

Homework 3: Symmetric and Public-Key Cryptography

Due: October 19, 2022 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/fa22/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: Hash-then-Encrypt [25 points]. The Android KeyStore uses "hash-then-CBC-encrypt" to construct an authenticated encryption scheme to generate and manage cryptographic keys for Android applications. Abstractly, the scheme operates as follows: Let $(\mathsf{Encrypt}_{\mathsf{CBC}}, \mathsf{Decrypt}_{\mathsf{CBC}})$ be a randomized CBC-mode encryption scheme built from a block cipher $F \colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$. Let $H \colon \mathcal{X}^{\le L} \to \mathcal{X}$ be a collision-resistant hash function. Define the following candidate authenticated encryption scheme $(\mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Encrypt}(k, m)$: Output $c \leftarrow \mathsf{Encrypt}_{\mathsf{CBC}}(k, H(m) \| m)$.
- $\mathsf{Decrypt}(k, c)$: Compute $(t, m) \leftarrow \mathsf{Decrypt}_{\mathsf{CBC}}(k, c)$ and output m if $t = H(m)$ and $\perp$ otherwise.

In the following, assume that $\mathcal{X} = \{0,1\}^n$ and $L \ge 2$.

(a) Show that $(\mathsf{Encrypt}, \mathsf{Decrypt})$ does not provide ciphertext integrity.

(b) Show that $(\mathsf{Encrypt}, \mathsf{Decrypt})$ is not CCA-secure. Recall that for encryption schemes over a variable-length message space, the adversary can only query the encryption oracle on pairs (m_0, m_1) where m_0 and m_1 have the *same* length.

(c) Would the above problems go away if the Android KeyStore had used randomized counter mode encryption instead of CBC-mode encryption? Give a brief explanation.

Both attacks show that the Android KeyStore does not provide authenticated encryption. These attacks were discovered in January 2016 and Google has confirmed that the encryption scheme will be removed from the system.

Problem 2: Goldreich-Levin and List Decoding [35 points]. Fix a vector $x \in \mathbb{Z}_2^n$. For a vector $r \in \mathbb{Z}_2^n$, let $\langle x, r \rangle = \sum_{i \in [n]} x_i r_i \bmod 2$ be the inner product between x and r. Suppose $\mathcal{A}$ is an efficient *deterministic* algorithm that takes as input r and predicts the value of $\langle x, r \rangle$. We say that $\mathcal{A}$ succeeds with probability p if

$$\Pr\left[r \xleftarrow{\text{R}} \mathbb{Z}_2^n : \mathcal{A}(r) = \langle x, r \rangle \right] \ge p.$$

In lecture, we showed that when $p \geq 3/4 + \varepsilon$ for some constant $\varepsilon > 0$, we can construct an algorithm $\mathcal{B}$ that recovers x with probability at least 9/10 by querying algorithm $\mathcal{A}$ on $O(n)$ inputs. We often call $\mathcal{B}$ 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 deterministic algorithm $\mathcal{A}$ such that

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

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 if algorithm $\mathcal{A}$ only succeeds in predicting $\langle x, r\rangle$ with probability $p = 1/2 + \varepsilon$, then it is not generally possible to use $\mathcal{A}$ to construct $\mathcal{B}$ that recovers $x \in \mathbb{Z}_2^n$ with probability better than 1/2. Instead, we relax our requirement to allow $\mathcal{B}$ to output a list $L \subset \mathbb{Z}_2^n$ of *possible* x's, and we say that $\mathcal{B}$ succeeds if $x \in L$ and $|L| = \text{poly}(n)$. We refer to $\mathcal{B}$ as a "list decoder."

(b) Suppose that in addition to algorithm $\mathcal{A}$, you have access to a second "magic" oracle $\mathcal{O}_x$. The oracle $\mathcal{O}_x$ takes no input, and each time you query it, it samples a fresh $r \xleftarrow{\text{R}} \mathbb{Z}_2^n$ and outputs the pair $(r, \langle x, r\rangle)$. Show how to use algorithm $\mathcal{A}$ and oracle $\mathcal{O}_x$ to construct an efficient algorithm $\mathcal{C}$ where for *all* inputs $y \in \mathbb{Z}_2^n$,

$$\Pr\left[\mathcal{C}(y) = \langle x, y\rangle\right] \geq 1 - 1/(10n).$$

Your algorithm $\mathcal{C}$ is allowed to make at most $O(\log n)$ queries to the magic oracle $\mathcal{O}_x$.

(c) Suppose now that you no longer have access to the magic oracle $\mathcal{O}_x$. Show that nonetheless, you can use the algorithm $\mathcal{C}$ you constructed above to build an efficient *list decoder* $\mathcal{B}$. Your algorithm $\mathcal{B}$ can run algorithm $\mathcal{A}$ on poly(n) inputs and should output a list $L \subset \mathbb{Z}_2^n$ of size poly(n) such that $x \in L$ with probability at least 9/10.

Note that the algorithm $\mathcal{B}$ no longer has access to the magic oracle $\mathcal{O}_x$. Algorithm $\mathcal{B}$ can still run the algorithm $\mathcal{C}$ you constructed above, but if it does so, it will need a way to answer $\mathcal{C}$'s queries to $\mathcal{O}_x$. **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 $2^k = \text{poly}(n)$.

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 3: DDH in Composite-Order Groups [15 points]. Let $k > 1$ be a constant, and let $\mathbb{G}$ be a cyclic group of order kq. 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 $a, b, r \xleftarrow{\text{R}} \mathbb{Z}_{kq}$) 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 4: Time Spent [2 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?