CS 346: Cryptography **Fall 2024**

Homework 4: Public-Key Cryptography

Due: November 12, 2024 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/fa24/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: Commitment Schemes from Discrete Log [18 points]. A commitment scheme is a digital analog of a "sealed envelope." Specifically, a sender can *commit* to a message m and send the resulting commitment c to a receiver (i.e., seal the message in an envelope). The commitment c should not reveal anything about the committed value m. Later on, the sender can *open* up the commitment and convince the receiver that c is indeed a commitment to the message m (i.e., open up the envelope and recover the original message). The commitment scheme is *hiding* if c hides the message m and is *binding* if the sender cannot open the commitment c to any message $m' \neq m$. In this problem, we will construct a commitment scheme from the discrete log assumption:

- **Public parameters:** Let $\mathbb{G}$ be a group of prime order p and let $g, h \in \mathbb{G}$ be arbitrary elements of $\mathbb{G}$ (that are not the identity element).
- **Commitment:** To commit to a message $m \in \mathbb{Z}_p$, sample $r \xleftarrow{R} \mathbb{Z}_p$ and output the commitment $c \leftarrow g^m h^r$.
- **Open:** To open the commitment c to the message m, the sender gives (m, r) to the receiver and the receiver checks that $c = g^m h^r$.

(a) Show that the above commitment scheme is *perfectly hiding* (i.e., the commitment c does not leak *any* information about the committed message m). Namely, show that given the commitment $c \in \mathbb{G}$, every candidate message $m' \in \mathbb{Z}_p$ is *equally likely* (over the randomness of r). One way to show this is that for every $m' \in \mathbb{Z}_p$, there is a *unique* $r' \in \mathbb{Z}_p$ such that $c = g^{m'} h^{r'}$.

(b) Show that the above commitment scheme is *computationally binding* assuming hardness of discrete log in $\mathbb{G}$. Namely, show that if an efficient adversary can output a commitment c together with openings (m, r) and (m', r') such that $g^m h^r = c = g^{m'} h^{r'}$ and $m \neq m'$, then the adversary can also compute the discrete log of h base g. In other words, if the sender can open the commitment in two different ways, then it can also compute the discrete log of h in $\mathbb{G}$.

Remember to give a brief explanation why any inverses you take actually exist.

Problem 2: DDH in Composite-Order Groups [14 points]. Let $\mathbb{G}$ be a cyclic group of order $2q$ where q is odd. 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 $a, b, r \xleftarrow{\text{R}} \mathbb{Z}_{2q}$) are indistinguishable to any efficient adversary.)

Remark: This shows that the DDH assumption does not hold over $\mathbb{Z}_p^*$ whenever $p = 2q + 1$. 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: Encrypted Group Chat [28 points]. Suppose a group of n people (denoted $P_1, \dots, P_n$) 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 $s \xleftarrow{\text{R}} \mathbb{Z}_p$. 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 $r_i \xleftarrow{\text{R}} \mathbb{Z}_p$ 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 $P_1, \dots, P_n$ 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 $\mathcal{A}$,

$$\Pr[\mathcal{A}(x_2, x_2^s, \dots, x_n, x_n^s) = g^s] = \text{negl}(\lambda), \tag{1}$$

where $x_i = g^{r_i}$ and $r_2, \dots, r_n, s \xleftarrow{\text{R}} \mathbb{Z}_p$. 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. (1) 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 $(x_2, x_2^s, \dots, x_n, x_n^s)$, then there exists an efficient algorithm $\mathcal{B}$ that breaks CDH in $\mathbb{G}$. Your algorithm should work for any polynomially-bounded n (i.e., you should *not* fix a value of n in your reduction) **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 4: Time Spent [1 point]. 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?