CS 388H: Cryptography **Fall 2022**

Homework 4: Public-Key Cryptography

Due: November 2, 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 Functions from Discrete Log [20 points]. Let $\mathbb{G}$ be a group of prime order p with generator g. Sample $h_1, \dots, h_n \xleftarrow{R} \mathbb{G}$ and define the hash function $H_{h_1,\dots,h_n} \colon \mathbb{Z}_p^n \to \mathbb{G}$ as follows:

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

(a) Show that $H_{h_1,\dots,h_n}$ 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 $h_1, \dots, h_n \xleftarrow{R} \mathbb{G}$ and it succeeds if it outputs $(x_1,\dots,x_n) \neq (x_1',\dots,x_n')$ such that $H_{h_1,\dots,h_n}(x_1,\dots,x_n) = H_{h_1,\dots,h_n}(x_1',\dots,x_n')$. 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 $H_{h_1,\dots,h_n}$ has a *trapdoor* that can be used to sample pre-images. Specifically, show that if someone knew the discrete logs of $h_1, \dots, h_n$ (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 $(x_1,\dots,x_n) \in \mathbb{Z}_p^n$ such that $H_{h_1,\dots,h_n}(x_1,\dots,x_n) = g^t$.

Problem 2: Encrypted Group Chat [30 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 $s \xleftarrow{R} \mathbb{Z}_p$. 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 $P_1, \dots, P_n$ 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 $\mathcal{A}$,

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

where $x_i = g^{r_i}$ and $r_2, \dots, r_n, s \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. 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 3: Computing on Encrypted Data [25 points]. Let $N = pq$ be an RSA modulus and suppose that $\gcd(N, \varphi(N)) = 1$. Consider the following public-key encryption scheme with message space $\mathbb{Z}_N$. The public key $\mathsf{pk} = N$ is the RSA modulus $N = pq$ and the secret key sk is the factorization $\mathsf{sk} = (p, q)$. Let $g = 1 + N \in \mathbb{Z}_{N^2}^*$. To encrypt a message $m \in \mathbb{Z}_N$, sample $h \xleftarrow{\mathrm{R}} \mathbb{Z}_{N^2}^*$ and compute $c \leftarrow g^m h^N \in \mathbb{Z}_{N^2}^*$.

(a) Show that the discrete logarithm assumption base g in $\mathbb{Z}_{N^2}^*$ is easy. Namely, give an efficient algorithm that takes as input (g, h) where $h = g^x$ for some $x \in \mathbb{Z}_N$, and outputs x. Remember to be explicit about the existence of any inverses you use in your solution. **Hint:** Use the binomial theorem: $(a+b)^k = \sum_{i=0}^k \binom{k}{i} a^i b^{k-i}$.

(b) Show how to *efficiently* implement the decryption algorithm $\mathsf{Decrypt}(\mathsf{sk}, c)$. Namely, describe an efficient algorithm that given the secret key $\mathsf{sk} = (p, q)$ and a ciphertext $c = g^m h^N$, outputs the message $m \in \mathbb{Z}_N$. You may use the fact that $\varphi(N^2) = N\varphi(N)$.

(c) Show that this public-key encryption scheme is semantically secure assuming that no efficient adversary is able to distinguish the following two distributions:

$$(N, u) \quad \text{and} \quad (N, v),$$

where $N = pq$ is an RSA modulus, $u \xleftarrow{\mathrm{R}} \mathbb{Z}_{N^2}^*$ and $v \xleftarrow{\mathrm{R}} \{h \in \mathbb{Z}_{N^2}^* : h^N\}$. Namely, show that the above encryption scheme is semantically secure assuming that it is hard to distinguish random values in $\mathbb{Z}_{N^2}^*$ from random N^{th} powers in $\mathbb{Z}_{N^2}^*$.

(d) Show that given the public key pk and two ciphertexts $c_1 \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_1)$, $c_2 \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_2)$, there is an efficient algorithm that outputs a new ciphertext c where $\mathsf{Decrypt}(\mathsf{sk}, c) = m_1 + m_2 \in \mathbb{Z}_N$. Your algorithm should only depend on *public* parameters and *not* the value of the messages m_1, m_2.

Remark: This is an example of an encryption scheme that supports computation on *encrypted* values.

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?