1 Fun with Collision-Resistant Hash Functions

We say that \((\text{ParamGen}, \text{Hash})\) is a family of collision-resistant hash functions (CRHFs) if the following two properties hold:

**Collision resistance:** For all p.p.t. \(A\), there exists a negligible \(\nu\) such that:

\[
\Pr[\text{params} \leftarrow \text{ParamGen}(1^k); (x, y) \leftarrow A(1^k, \text{params}) : x \neq y \text{ and } \text{Hash}_{\text{params}}(x) = \text{Hash}_{\text{params}}(y)] \leq \nu(k)
\]

**Length reduction:** The function \(\text{Hash}\) is length-reducing, meaning that for all \(\text{params}\) in the output set of \(\text{ParamGen}(1^k)\),

\[
\text{Hash}_{\text{params}} : \{0, 1\}^* \rightarrow \{0, 1\}^k
\]

Once \(\text{params}\) is fixed, let \(H\) denote \(\text{Hash}_{\text{params}}\). Let \(H_1\) and \(H_2\) be length-reducing functions.

a. Use \(H\) to construct a hash function \(H_a\) over the same domain and range that is also collision-resistant, but such that if you truncate the least significant bit (LSB) of the output of \(H_a\), then this new hash function is no longer collision-resistant. More formally, let \(H_a(x) = z_1z_2\cdots z_{k-1}z_k\) and let \(H_a'(x) = z_1z_2\cdots z_{k-1}\). Design \(H_a\) such that \(H_a\) is collision-resistant but \(H_a'\) is not.

b. Next, suppose we want to build a CRHF \(H_b\) using \(H_1\) and \(H_2\) such that if at most one of \(H_1\) or \(H_2\) is not collision-resistant, then \(H_b\) is still a CRHF. Define:

\[
H_b(x) = (H_1(x), H_2(x))
\]

Show that \(H_b\) is a CRHF if either \(H_1\) or \(H_2\) is a CRHF.

c. Show that \(H_c(x) = H_1(H_2(x))\) might not be a CRHF even if one of \(H_1\) or \(H_2\) is a CRHF.

d. Let \(F\) be a PRF. Prove that \(F_d(s, x) = F(s, H(x))\) is a PRF. This shows that a CRHF can be used to extend the domain of a PRF.
2 Adaptive Security

Recall the definition of semantic security for public-key encryption:

**Definition 1 (Semantic security)** A cryptosystem $(\text{KeyGen}, \text{Enc}, \text{Dec})$ is **semantically secure** if there exists a p.p.t. simulator $\text{Sim}$ such that for all p.p.t. eavesdroppers $\text{Eve}$, the following two distributions are computationally indistinguishable:

$$\text{Real}(1^k) = \{(pk, sk) \leftarrow \text{KeyGen}(1^k); (m, state) \leftarrow \text{Eve}(1^k, pk); c \leftarrow \text{Enc}(pk, m); \text{view} \leftarrow \text{Eve}(state, c) : \text{view}\}$$

$$\text{Simulated}(1^k) = \{(pk, sk) \leftarrow \text{KeyGen}(1^k); (m, state) \leftarrow \text{Eve}(1^k, pk); c \leftarrow \text{Sim}(pk, |m|); \text{view} \leftarrow \text{Eve}(state, c) : \text{view}\}$$

This definition is **adaptive**, meaning that on input the public key, $\text{Eve}$ adaptively chooses a message for which she thinks she can distinguish a real ciphertext from a meaningless one created by a simulator.

Now consider a different approach where we require that security hold for all messages rather than ones adaptively selected by $\text{Eve}$. To make sure that the length of the message can be a function of the security parameter, we quantify not over all messages, but over all sequences of messages $M$ where the length of the $k^{th}$ message $m_k$ is at most $q(k)$ for some polynomial $q$. Let us call such a sequence $M$ a **poly-length message set**.

**Definition 2 (Non-adaptive semantic security)** A cryptosystem $(\text{KeyGen}, \text{Enc}, \text{Dec})$ is **non-adaptively semantically secure** if there exists a p.p.t. simulator $\text{Sim}$ such that for all poly-length message sets $M = \{m_k\}_{k=1}^\infty$, for all p.p.t. eavesdroppers $\text{Eve}$, the following two distributions are computationally indistinguishable:

$$\text{Real}(1^k) = \{(pk, sk) \leftarrow \text{KeyGen}(1^k); c \leftarrow \text{Enc}(pk, m_k); \text{view} \leftarrow \text{Eve}(pk, c, m_k) : \text{view}\}$$

$$\text{Simulated}(1^k) = \{(pk, sk) \leftarrow \text{KeyGen}(1^k); c \leftarrow \text{Sim}(pk, |m_k|); \text{view} \leftarrow \text{Eve}(pk, c, m_k) : \text{view}\}$$

In this problem, you will show how these two definitions are related to each other.

a. Suppose that $(\text{KeyGen}, \text{Enc}, \text{Dec})$ is a cryptosystem satisfying both Definition 1 and Definition 2. Show how you can modify it to construct a contrived cryptosystem $(\text{KeyGen}^*, \text{Enc}^*, \text{Dec}^*)$ that satisfies Definition 2 but not Definition 1.

b. Must a cryptosystem that satisfies Definition 1 also satisfy Definition 2?

For this problem, it is important to keep in mind that the adversary is typically modelled as a non-uniform Turing machine. That is, the adversary invoked for security parameter $1^k$ may have a poly-length string (say, the message $m_k$) hard-wired into its description.
3 Nested Encryption

Let \((\text{KeyGen}, \text{Enc}, \text{Dec})\) be a semantically secure public-key encryption scheme. Define a new \textit{nested} encryption scheme \((\text{KeyGen'}, \text{Enc'}, \text{Dec'})\) as follows:

- **Key generation:** \(\text{KeyGen}'(1^k)\) runs \(\text{KeyGen}(1^k)\) twice to obtain \((pk_1, sk_1)\) and \((pk_2, sk_2)\). It outputs \((pk = (pk_1, pk_2), sk = (sk_1, sk_2))\).
- **Encryption:** \(\text{Enc}'(m, pk; r) = \text{Enc}(\text{Enc}(m, pk_1; r_1), pk_2; r_2)\), where \(r = r_1 \circ r_2\).
- **Decryption:** \(\text{Dec}'(c, sk) = \text{Dec}(\text{Dec}(c, sk_2), sk_1)\).

Prove that the nested encryption scheme \((\text{KeyGen'}, \text{Enc'}, \text{Dec'})\) is also a semantically secure PKE scheme. In particular, provide a reduction \(B\) such that if \(A\) can break the nested encryption scheme, then \(B\) can break the original scheme.

4 Cramer-Shoup Cryptosystem

In this problem we will discuss the Cramer-Shoup cryptosystem, which we saw partially in class. Here is our first attempt at defining the cryptosystem:

- **Setup:** Both parties know the group \(\mathbb{G}\), its prime order \(q\), and a generator \(g\).
- **Key generation:** The keys are generated as follows: First pick \(x, y, a, b \leftarrow \mathbb{Z}_q\) and retain these values as the secret key. The public key then consists of \(g, h, A = g^x h^y\), and \(B = g^a h^b\), where \(h = g^\alpha\) for \(\alpha \leftarrow \mathbb{Z}_q\). (Note that \(h\) is also a generator of \(\mathbb{G}\).)
- **Encryption:** Given a message \(m \in \mathbb{G}\), first pick a random value \(r \leftarrow \mathbb{Z}_q\). Output the tuple \((g^r, h^r, A^r \cdot m, B^r)\).
- **Decryption:** Given the tuple \((R, S, P, T)\), first check that \(T = R^a S^b\). If this check passes, output \(P \frac{R}{R^r S^y}\). Otherwise, output fail.

As we saw in class, the security of the Cramer-Shoup cryptosystem is based on the DDH assumption:

**Assumption 1 (DDH)** Let \(\text{SafePrime}\) be an algorithm which, on input \(1^k\), outputs integers \(p\) and \(q\) such that \(p\) and \(q\) are both prime, \(p\) is \(k\) bits long, and \(p = 2q + 1\). We assume that the following distributions are computationally indistinguishable:

\[
\begin{align*}
D_0(1^k) &= \{(p, q) \leftarrow \text{SafePrime}(1^k); g \leftarrow QR_p; x \leftarrow \mathbb{Z}_q; y \leftarrow \mathbb{Z}_q; (p, g, g^x, g^y, g^{xy})\} \\
D_1(1^k) &= \{(p, q) \leftarrow \text{SafePrime}(1^k); g \leftarrow QR_p; x \leftarrow \mathbb{Z}_q; y \leftarrow \mathbb{Z}_q; z \leftarrow \mathbb{Z}_q; (p, g, g^x, g^y, g^z)\}
\end{align*}
\]
a. It turns out that this “reduced” version of the Cramer-Shoup cryptosystem is CCA1-secure, but it is not CCA2-secure. Prove the latter; that is, construct an adversary $A$ performing an adaptive chosen-ciphertext attack that wins the CCA2 game with nonnegligible probability. Recall that the CCA2 security experiment is described in problem 2 of HW8.

b. To make the reduced version more secure, we will need to make some modifications to the cryptosystem. Consider the new version:

- **Setup**: Both parties know the group $G$, its prime order $q$, and a generator $g$.
- **Key generation**: First pick $x, y, a, b, w, z \leftarrow \mathbb{Z}_q$ and retain these values as the secret key. The public key then consists of $g, h, A = g^x h^y, B = g^a h^b, C = g^w h^z$, and a collision-resistant hash function $H$.
- **Encryption**: Given a message $m \in G$, first pick a random value $r \leftarrow \mathbb{Z}_q$. Output the tuple $(g^r, h^r, A^r \cdot m, (BC\beta)^r)$, where $\beta = H(g^r, h^r, A^r \cdot m)$.
- **Decryption**: Given the tuple $(R, S, P, T)$, first check that $T = R^{\alpha + \beta w} S^{b + \beta z}$. If this check passes, output $\frac{P}{R^S T}$. Otherwise, output fail.

Verify that this cryptosystem is correct; that is, given a valid ciphertext, the decryption algorithm correctly computes the appropriate plaintext message.