CS 276: Homework 3

Due Date: Sunday, Mar 15, 2026 at 8:59pm via Gradescope

Usage of LLMs/Generative AI tools is prohibited. Other online resources (textbooks/lecture notes) are permissible.

Definition 0.1 (Trapdoor Permutation (TDP)) *A* trapdoor permutation *is a trapdoor function* $(\mathsf{TDP.Gen}, f, f^{-1})$ *where for every* $(s,t) \leftarrow \mathsf{TDP.Gen}(1^n)$*, the function* $f(s,\cdot) : \mathcal{D}(s) \to \mathcal{D}(s)$ *is a* permutation *on its domain* $\mathcal{D}(s)$*.*

Notation. To avoid confusion between the public-key and secret-key schemes, we write the secret-key encryption scheme as $\mathsf{SKE} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$, where G generates a key, E encrypts, and D decrypts.

1. **(IND-CCA Security from Trapdoor Permutations.)**

   Let $(\mathsf{TDP.Gen}, f, f^{-1})$ be a trapdoor permutation and let H be a random oracle $H : \{0,1\}^* \to \{0,1\}^n$. Let $\mathsf{SKE} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$ be an IND-CCA-secure secret-key encryption scheme. Consider the following public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

   - $\mathsf{Gen}(1^n)$: Run $\mathsf{TDP.Gen}(1^n)$ to obtain (s,t). Set $\mathsf{pk} = s$ and $\mathsf{sk} = t$.
   - $\mathsf{Enc}(\mathsf{pk}, m)$: Sample $x \xleftarrow{\$} \mathcal{D}(s)$ and compute $y \leftarrow f(s,x)$. Encrypt m using the secret-key scheme with key $H(x)$: compute $z \leftarrow \mathsf{E}(H(x), m)$. Output $c = (y,z)$.
   - $\mathsf{Dec}(\mathsf{sk}, c)$: Parse $c = (y,z)$ and compute $x \leftarrow f^{-1}(t,y)$. Decrypt using the secret-key scheme: compute $m \leftarrow \mathsf{D}(H(x), z)$. Output m.

   Prove that Π is IND-CCA-secure in the random oracle model.

   Hint: We have shown that this is IND-CPA-secure (Theorem 6.1)

   Solution **Proof.** We proceed by a sequence of hybrids. Let W_i denote the event that $\mathcal{A}$ outputs $b' = b$ in Hyb_i.

   Hyb_0. The standard IND-CCA game for Π. The challenger samples $(s,t) \leftarrow \mathsf{TDP.Gen}(1^n)$, sets $\mathsf{pk} = s$, and for the challenge samples $x^* \xleftarrow{\$} \mathcal{D}(s)$, sets $y^* = f(s,x^*)$, $k^* = H(x^*)$, and returns $c^* = (y^*, \mathsf{E}(k^*, m_b))$. $\mathcal{A}$'s advantage is $|\Pr[W_0] - 1/2|$.

   Hyb_1. Replace $k^* = H(x^*)$ with an independent uniform key $k^* \xleftarrow{\$} \{0,1\}^n$ used to compute the challenge ciphertext $c^* = (y^*, \mathsf{E}(k^*, m_b))$. Decryption queries with $y = y^*$ also use this same key k^*. The random oracle H functions normally. Let Ask be the event that $\mathcal{A}$ queries $H(x^*)$. Until Ask occurs, k^* is uniformly random and perfectly hidden from $\mathcal{A}$'s view, so Hyb_0 and Hyb_1 are identical. Thus $|\Pr[W_0] - \Pr[W_1]| \leq \Pr[\mathsf{Ask}]$.

   Bounding $\Pr[\mathsf{Ask}]$ via TDP one-wayness. Given challenge (s, y^*), adversary $\mathcal{B}_{TDP}$ sets $\mathsf{pk} = s$, samples random k^*, b, and returns $c^* = (y^*, \mathsf{E}(k^*, m_b))$. It simulates the random oracle H lazily, bypassing the need for x by maintaining a table T_{KEY} mapping $y \in \mathcal{D}(s)$ to $\{0,1\}^n$:

- **On query $H(q)$:** compute $y' = f(s, q)$. If $y' = y^*$, it outputs q and halts (successfully inverting y^*). Otherwise, if $y' \notin T_{KEY}$, it samples $k \xleftarrow{\$} \{0,1\}^n$ and sets $T_{KEY}[y'] = k$. It returns $T_{KEY}[y']$.
- **On decryption query $(y, z) \neq c^*$:** if $y = y^*$, return $\mathsf{D}(k^*, z)$. If $y \neq y^*$: check if $y \notin T_{KEY}$; if so, sample $k \xleftarrow{\$} \{0,1\}^n$ and set $T_{KEY}[y] = k$. Return $\mathsf{D}(T_{KEY}[y], z)$.

Because $f(s, \cdot)$ is a permutation, the mapping between x and y is a bijection, making this simulation mathematically identical to standard lazy evaluation of $H(x)$. This perfectly simulates Hyb_1 without using t up to the point Ask occurs, so $\Pr[\mathsf{Ask}] \leq \mathsf{negl}(n)$.

Bounding $|\Pr[W_1] - 1/2|$ via SKE IND-CCA. Conditioned on Ask not occurring, $\mathcal{A}$ never queries $H(x^*)$, so the key k^* is used only through the SKE scheme. Adversary $\mathcal{B}_{SKE}$ generates (s, t), samples x^*, sets $y^* = f(s, x^*)$, and gives $\mathsf{pk} = s$ to $\mathcal{A}$. It simulates H normally (lazily). If $\mathcal{A}$ queries $H(x^*)$, it aborts (corresponding to Ask). On challenge (m_0, m_1), it forwards them to its SKE challenger, receives z^*, and returns $c^* = (y^*, z^*)$. For decryption query $(y, z) \neq c^*$: if $y = y^*$, forward z to the SKE decryption oracle; if $y \neq y^*$, use t to compute $x = f^{-1}(t, y)$, evaluate $H(x)$ (lazily sampling if not yet queried), and decrypt using this key by returning $\mathsf{D}(H(x), z)$. This perfectly simulates Hyb_1 conditioned on $\neg\mathsf{Ask}$. Thus $|\Pr[W_1] - 1/2| \leq \mathsf{negl}(n)$.

Combining: $|\Pr[W_0] - 1/2| \leq |\Pr[W_0] - \Pr[W_1]| + |\Pr[W_1] - 1/2| \leq \mathsf{negl}(n)$. ■