Problem 1

Let \((G, E, D)\) be a secure public-key encryption scheme. Define the pair \((S, R)\) as follows:

\[
S(1^k, x) \equiv \{(PK, SK) \leftarrow G(1^k) ; \; z \leftarrow E(PK, x) ; \; c \leftarrow (PK, z) ; \; d \leftarrow SK : (c, d)\},
\]

\[
R(c, x, d) \equiv \begin{cases} 
1 & \text{if } D(SK, z) = x \\
0 & \text{otherwise}
\end{cases}.
\]

Prove or disprove that the fact that \((S, R)\) is a string commitment scheme. (If it is, state whether its hiding and binding properties are computational or perfect.)

**Solution:** We prove that \((S, R)\) is NOT a commitment scheme. Let \((G', E', D')\) be a secure public-key encryption scheme. Using \((G', E', D')\), we construct the triple \((G, E, D)\) as follows:

- \(G = G'\)
- \(E = E'\)
- The decryption algorithm \(D\), on input \((SK, z)\), outputs 0\(^k\) if \(SK = 0^k\), and outputs \(D'(SK, z)\) otherwise.

By the security of \((G', E', D')\), the probability that the key generator \(G'\) outputs a key pair \((PK, SK)\) with \(SK = 0^k\) is negligible. Hence, for almost all outputs \((PK, SK)\) of \(G\), \(E(PK, \cdot) = E'(PK, \cdot)\) and \(D(SK, \cdot) = D'(SK, \cdot)\), so that \((G, E, D)\) inherits the security of \((G', E', D')\).

On the other hand, for the particular secure public-key encryption scheme \((G, E, D)\), \((S, R)\) is not computationally binding. Indeed, consider the malicious sender \(\bar{S}\) that, on input \(1^k\), runs the key generator \(G(1^k)\) to obtain the key pair \((PK, SK)\) and outputs the following 5-tuple:

\[
(c = E(PK, 1^k), m_0 = 0^k, d_0 = 0^k, m_1 = 1^k, d_1 = SK) .
\]

So the receiver \(R\) is fooled, because it accepts both \((c, m_0, d_0)\) and \((c, m_1, d_1)\).

**Remark.** While the above counterexample works, it almost feels like “cheating” — any reasonable \(R\) that knows about the construction for \((S, R)\) would get suspicious upon seeing a decommitment \(SK = 0^k\). In fact, \(R\) does not even attempt to figure out whether \((PK, SK)\) is a valid key-pair by encrypting and decrypting some arbitrary message. On the other hand, \(R\) as constructed above is the definition of the receiver’s reasoning for figuring out whether to accept a decommitment or not, and any counterexample is valid as long as it works for the above \(R\).

Nonetheless, we can provide a “more powerful” counterexample, which will fool not only the receiver \(R\) defined above, but any probabilistic polynomial-time receiver, thus demonstrating that the “encryption” construction above is inherently flawed. Unfortunately, this other counterexample is slightly harder to prove correct, so we will only sketch its hybrid argument.
Let \((G', E', D')\) be a secure public-key encryption scheme (with perfect completeness). Using \((G', E', D')\), we construct the triple \((G, E, D)\) as follows:

\[
G(1^k) \equiv \{(PK_0, SK_0) \leftarrow G'(1^k) : (PK_1, SK_1) \leftarrow G'(1^k) : b \leftarrow \{0,1\} : ((PK_0, PK_1), (b, SK_0))\},
\]

\[
E(PK, m) \equiv \{c_0 \leftarrow E'(PK_0, m) ; c_1 \leftarrow E'(PK_1, m) ; (c_0, c_1)\},
\]

\[
D(SK, c) \equiv D'(SK_b, c_b).
\]

We claim that \((G, E, D)\) is GM-secure. For \(k \in \mathbb{N}\) and \(m_0, m_1 \in \{0,1\}^k\), define the following hybrid distributions:

\[
H^0_k \equiv \{(PK_0, SK_0) \leftarrow G'(1^k) ; (PK_1, SK_1) \leftarrow G'(1^k) ; c_0 \leftarrow E'(PK_0, m_0) ;
\]

\[
c_1 \leftarrow E'(PK_1, m_0) : ((c_0, c_1), (PK_0, PK_1))\},
\]

\[
H^1_k \equiv \{(PK_0, SK_0) \leftarrow G'(1^k) ; (PK_1, SK_1) \leftarrow G'(1^k) ; c_0 \leftarrow E'(PK_0, m_0) ;
\]

\[
c_1 \leftarrow E'(PK_1, m_1) : ((c_0, c_1), (PK_0, PK_1))\},
\]

\[
H^2_k \equiv \{(PK_0, SK_0) \leftarrow G'(1^k) ; (PK_1, SK_1) \leftarrow G'(1^k) ; c_0 \leftarrow E'(PK_0, m_1) ;
\]

\[
c_1 \leftarrow E'(PK_1, m_1) : ((c_0, c_1), (PK_0, PK_1))\}.
\]

By the security of \((G', E', D')\), we deduce that \(H^2_k \approx H^1_k\) and \(H^1_k \approx H^0_k\). Therefore, \(H^0_k \approx H^2_k\). However, \(H^0_k\) and \(H^2_k\) are exactly the distributions that an adversary is asked to distinguish when attacking the GM-security of \((G, E, D)\). Thus, we conclude that \((G, E, D)\) is indeed GM-secure.

On the other hand, for the particular secure public-key encryption scheme \((G, E, D)\), \((S, R)\) is not computationally binding, because a cheating sender could commit as in \(H^1_k\) and then de-commit in two ways by sending as a de-commitment either the secret key \((0, SK_0)\) or the secret key \((1, SK_1)\).

**Problem 2**

Prove that commitment schemes that are both perfectly hiding and perfectly binding do not exist.

**Solution:** Suppose that \((S, R)\) is perfectly binding. That means that for each \(c \in \{0,1\}^*\) there exists at most one \(x \in \{0,1\}^*\) such that there exists a \(d \in \{0,1\}^*\) for which \(R(c, x, d) = 1\). Given some commitment \(c \in \{0,1\}^*\) to some string, a computationally unbounded adversary can simply try, one by one and for each \(k \in \mathbb{N}\), all the pairs \((x, d)\) in \(\bigcup_{i \leq k} \{0,1\}^i \times \{0,1\}^j\) and check if \(R(c, x, d) = 1\). The perfect binding property ensures that such an adversary will find (in finite time) the unique \(x\) and some \(d\) such that \(R(c, x, d) = 1\). We must conclude that \((S, R)\) is not perfectly hiding.

**Problem 3**

**Definition 1.** Let \(f_0, f_1\) be polynomial-time computable, injective and length-preserving functions from \(\{0,1\}^*\) to \(\{0,1\}^*\). We say that \((f_0, f_1)\) are claw-free permutations, if \(\forall PPTA, \forall c > 0, \forall s.I.\ k,\)

\[
Pr[(x_0, x_1) \leftarrow A(1^k) : f_0(x_0) = f_1(x_1)] < k^{-c}.
\]

**Definition 2.** Let \(H\) be a sequence of functions, \(H = \{H_k\}_{k=1,2,\ldots}, H_k : \{0,1\}^* \to \{0,1\}^k\), such that there exists a polynomial-time computable function \(f(\cdot, \cdot)\) such that \(\forall k > 0, \forall x \in \{0,1\}^*, f(1^k, x) = H_k(x)\). We say that \(H\) is a family of collision-resistant hash functions, if \(\forall PPT B, \forall c > 0, \forall s.I. k,\)

\[
Pr[(a, b) \leftarrow B(1^k) : (a \not= b) \land (H_k(a) = H_k(b))] < k^{-c}.
\]
Prove that if claw-free permutations exist, then so do collision-resistant hash families.

**Solution.** Given any \( a \in \{0,1\}^* \), we denote \( \bar{a} \) as a prefix-free encoding of \( a \) (say, Huffman encoding or the sort). \( \forall k \), given \( a \in \{0,1\}^* \), let \( \bar{a} = a_1 \ldots a_n \) for some \( n > 0 \), here is how \( H_k \) works. Let \( c_{n}(\bar{a}) = 0^k \). For \( j \) from \( n - 1 \) to 0, define \( c_j(\bar{a}) = f_0(c_{j+1}(\bar{a})) \) if \( \bar{a}_{j+1} = 0 \), and \( c_j(\bar{a}) = f_1(c_{j+1}(\bar{a})) \) if \( \bar{a}_{j+1} = 1 \). Let \( H_k(a) = c_0(\bar{a}) \). We claim that \( H \) is collision resistant. Suppose not, i.e., there is a PPT \( B \) such that for infinitely many \( k \)'s, \( B \) finds a collision \((a,b)\) for \( H_k \) with non-negligible probability, then for all those \( k \)'s, given \( 1^k \), we also find a claw \((x_0,x_1)\) for \( f_0,f_1 \) with the same probability. Indeed, take the smallest \( j \) such that \( \bar{a}_j = \bar{b}_j \) (it exists because \( a \neq b \), and \( \bar{a} \) and \( \bar{b} \) are prefix-free).

Wlog assume that \( \bar{a}_j = 0 \) and \( \bar{b}_j = 1 \). As \( H_k(a) = c_0(\bar{a}) = H_k(b) = c_0(\bar{b}) \), \( a_{1\ldots(j-1)} = \bar{b}_{1\ldots(j-1)} \) and \( f_0 \) and \( f_1 \) are both permutations over \( \{0,1\}^k \), we know that \( c_{j-1}(\bar{a}) = c_{j-1}(\bar{b}) \). Let \( z = c_{j-1}(\bar{a}) \), \( x_0 = c_j(\bar{a}) \) and \( x_1 = c_j(\bar{b}) \), then \( z = f_0(x_0) = f_1(x_1) \), and we find a claw.

Note that it is OK here to begin from \( 0^k \) instead of a random \( R \), as this doesn’t change the way how we find a claw from any collision (if you use a random \( R \), make sure that \( R \) is also put into the index of \( H_k \) and it is fixed for the same \( H_k \)). In this way, seems that \( H_k \) is a single function and \( H \) can be constructed uniformly. However, notice that \( f_0,f_1 \) must be actually randomly chosen from a large family (otherwise we can just hard-wire a claw in a circuit), it follows that \( H_k \) is also randomly chosen from a family.

**Problem 4**

Let \((G,S,V)\) be a GMR-secure signature scheme where \( S \) is deterministic. Suppose that \( |SK| = k \) where \((PK,SK) \leftarrow G(1^k) \), and \( \forall SK,m \in \{0,1\}^k \), \( |S_{SK}(m)| = \ell(k) = |S_{SK}(1^k)| \), i.e., the length of signature is fixed. Consider the function family \( \{f_{s_1,s_2} : \{0,1\}^{\ell(k)} \rightarrow \{0,1\}\}_{s_1,s_2} \), where \( s_1 \) is selected as \( SK \) according to \( G(1^k) \) and \( s_2 \leftarrow \{0,1\}^{\ell(k)} \), such that \( f_{s_1,s_2}(\alpha) = S_{s_1}(\alpha) \cdot s_2 \), where \( \cdot \) is the inner product modulo 2.

Prove that this function family is pseudorandom (although it is not length preserving).

**Solution:** Consider hybrid experiments such that in the \( i \)-th hybrid the first \( i \) queries are answered by a truly random Boolean function and the rest of the queries are answered by a uniformly distributed \( f_{s_1,s_2} \). (Note that it seems important to use this order of random versus pseudorandom answers.) It is easy to show that distinguishability of the first and the last hybrids implies distinguishability of the \( i \)-th and \( i+1 \)st hybrids for some \( i \). And this further implies that a PPT \( A \) has a non-negligible advantage in the following game. In this game, \( A \) first is asked to select \( \alpha \); next \( f_{s_1,s_2} \) is uniformly selected, and \( A \) is given \( s_2 \) as well as oracle access to \( S_{s_1} \) (but is not allowed to query \( \alpha \)) and is asked to guess \( f_{s_1,s_2}(\alpha) \) (or equivalently, to distinguish \( f_{s_1,s_2}(\alpha) \) from a truly random bit).

Note that the particular order (of random versus pseudorandom answers in the hybrids) allows \( A \) to generate the (corresponding) hybrid while playing this game properly. That is, \( A \) answers the first \( i \) queries at random, sets \( \alpha \) to equal the \( i+1 \)st query, uses the challenge bit value as the corresponding answer, and uses \( s_2 \) and the oracle \( S_{s_1} \) to answer the subsequent queries. It is also important that the game be defined such that \( s_2 \) is given only after \( A \) has selected \( \alpha \).

At this point, one can easily apply the proof of the existence of hard-core predicate, as done in class, to show that by using \( A \), we can forge a signature of \((G,S,V)\). See the proof of Theorem 1 in https://omereingold.files.wordpress.com/2014/10/mac.pdf.