CS178 F21 Final Examination

Michael Glushchenko, 9403890

December 10, 2021

Problem 1.

Given: A keyed hash function with the security parameter λ:

$$H = \{h_k : \{0,1\}^m \to \{0,1\}^n\}_{k \in \{0,1\}^\lambda},$$

where $m = m(\lambda)$ and $n = n(\lambda)$ are polynomial in λ.
For a fixed $f : \{0,1\}^m \to \{0,1\}^n$, we say H is f-*secure* if $\forall$ PPT $\mathcal{A}$,

$$\mathcal{P}[h_k(x) = f(x) : k \xleftarrow{\$} \{0,1\}^\lambda, x \leftarrow \mathcal{A}(1^\lambda, k)] \leq \text{negligible}(\lambda).$$

Part A.

All we have to do here is find an m, n, and $\mathcal{A}$ such that $\mathcal{A}$ can give us an x such that $h_k(x) = f_1(x)$, for a $f_1 : \{0,1\}^m \to \{0,1\}^n$. Go ahead and set f_1 to be any invertible function. The intuition is that we need an invertible function, so that there actually exists some clever way us to create this x we need. I think that if we made f_1 take the modulus of 2^{m-n} of the input, we would have good chance of $h_k(x) = f(x)$, I am simply not sure how to show it.

Part B.

Goal: Find f_2 such that H *is* f_2*-secure.*
Solution: Consider an f_2 constructed in the following way:

$$f_2 := \{h_{\mathbf{k'}} : \{0,1\}^m \to \{0,1\}^n\}_{\mathbf{k'} \in \{0,1\}^\lambda}.$$

That is, we set f_2 to be a keyed hash function after sampling our own key. The goal now is to show that H is f_2*-secure*. Suppose not. Then $\exists$ PPT $\mathcal{A}$ such that

$$\mathcal{P}[h_k(x) = f(x) : k \xleftarrow{\$} \{0,1\}^\lambda, x \leftarrow \mathcal{A}(1^\lambda, k)] = \epsilon(\lambda),$$

and $\epsilon(\lambda)$ is a non-negligible function; label the above statement as the probability that $\mathcal{A}$ wins; i.e., $\mathcal{A}$ wins if it can give us an x that would result in $f_2(x) = h_k(x)$ with non-negligible probability ϵ.
Naturally, there's two disjoint cases possible:

- <u>Case 1</u>: $k = k'$. If so, $f_2(x) = h_k(x)$ on every input $x \in \{0,1\}^m$, which means $\mathcal{A}$ would succeed 100% of the time given this case; however, the probability of the case happening is

$$\mathcal{P}[k = k'] = \frac{1}{2^\lambda} = \text{negligible}(\lambda).$$

- <u>Case 2</u>: $k \neq k'$. If so, $\mathcal{A}$ needs an x that maps to the same hash of two differently-keyed hash functions, i.e., $\mathcal{A}$ needs to invert one of the hash functions to perform the necessary task; hash function pre-image resistance tells us

$$\mathcal{P}[\mathcal{A}(h(x)) \to x \text{ s.t. h(x) = x}] = \text{negligible}(\lambda)$$

  So, the probability $\mathcal{A}$ can find the pre-image of at least one of the keyed hash functions is also negligible.

As a result, we have a contradiction:

$$\mathcal{P}[\mathcal{A} \text{ "wins" the game and outputs a valid x}]$$

$$= \mathcal{P}[\mathcal{A} \text{ "wins" in case 1}] + \mathcal{P}[\mathcal{A} \text{ "wins" in case 2}]$$

$$= \text{negligible} = \epsilon(\lambda),$$

when, in fact, we declared ϵ to be non-negligible.

Problem 2.

Given: An RSA encryption scheme (Gen, Enc, Dec) **without** padding.
Claim: (Gen, Enc, Dec) is subject to an attack where a PPT $\mathcal{A}$, without knowing m_0 or m_1, can find the encryption of $m_0 \cdot m_1$, where $\cdot$ is modular multiplication in $\mathbb{Z}_N$.
Solution: In RSA, the Enc algorithm is defined as follows:

$$Enc((N, e), m) = m^e \pmod{N}.$$

Given $m_0, m_1 \xleftarrow{\$} \{0,1\}^\lambda$ their ciphertexts are

$$ct_0 = m_0^e \pmod{N} \quad \& \quad ct_1 = m_1^e \pmod{N}.$$

But then consider a PPT $\mathcal{A}$ that takes in two ciphertexts, and outputs a new ciphertext:

$$\mathcal{A}(ct_1, ct_2) = ct_1 \cdot ct_2 = ct_1 ct_2 \pmod{N}$$
$$= (Enc((N, e), m_0)Enc((N, e), m_1)) \pmod{N}$$
$$= ((m_0^e \pmod{N})(m_1^e \pmod{N})) \pmod{N}$$
$$= ((m_0 \pmod{N})^e(m_1 \pmod{N})^e) \pmod{N}$$
$$= (((m_0 \pmod{N})(m_1 \pmod{N}))^e) \pmod{N}$$
$$= ((m_0 m_1 \pmod{N})^e) \pmod{N} = Enc((N, e), m_0 \cdot m_1)$$
$$\Rightarrow \mathcal{A}(ct_1, ct_2) \to Enc((N, e), m_0 \cdot m_1) = ct$$

Why is this classified as an attack on (Gen, Enc, Dec)?
Because the above statement shows a clear relation between how a change in some message will result in a change in that message's cipher, as well as the fact that $\mathcal{A}$ extracts previously-unknown information out of the interaction: it's the ciphertext ct. The adversary $\mathcal{A}$ can now be used to create a reduction $\mathcal{B}$ that successfully recovers encoded messages of (Gen, Enc, Dec), without knowledge of the private key (N, d) or the message m, in polynomial time (eventually with very good probability). $\mathcal{B}$ can be chosen to run a number of cipher-only attacks, including a chosen cipher attack, a frequency analysis attack, and more, based on the fact that $\mathcal{A}$ extracts brand-new information out of its interaction with the Enc oracle.

Problem 3.

Given: An public-key encryption scheme (Gen, Enc, Dec).

Part A.

Given: $\exists m_0, m_1$ with ciphertext distributions

$$\mathcal{D}_b := \{(pk, Enc(pk, m_b)) : (pk, sk) \leftarrow Gen(1^\lambda)\}, \quad b \in \{0, 1\},$$

such that $\forall$ unbounded $\mathcal{A}$, we have

$$\begin{gathered} |\mathcal{P}[1 \leftarrow \mathcal{A}(1^\lambda, (pk, ct)) : (pk, ct) \leftarrow \mathcal{D}_0] \\ -\mathcal{P}[1 \leftarrow \mathcal{A}(1^\lambda, (pk, ct)) : (pk, ct) \leftarrow \mathcal{D}_1]| \\ \leq \frac{1}{2} \end{gathered}$$

Claim: (Gen, Enc, Dec) *does not* satisfy correctness.
Proof: We know that correctness is satisfied if, $\forall m$,

$$\mathcal{P}[Dec(sk, Enc(pk, m)) = m : (pk, sk) \leftarrow Gen(1^\lambda)] = 1.$$

Suppose (Gen, Enc, Dec) does satisfy correctness. Given 1^λ, pk, and a ciphertext ct we will have our $\mathcal{A}$ iterate through picking a message, encoding it, and see if the encoding matches the ciphertext it received. Once $\mathcal{A}$ finds such an encoding, it'll output 1. Since $\mathcal{A}$ is *unbounded*, we know that the probability it distinguishes the ciphertext distributions of (any) pair of two arbitrary messages m_0 and m_1 should be exactly $1 \rightarrow$ a contradiction.

$$\begin{gathered} |\mathcal{P}[1 \leftarrow \mathcal{A}(1^\lambda, (pk, ct)) : (pk, ct) \leftarrow \mathcal{D}_0] \\ -\mathcal{P}[1 \leftarrow \mathcal{A}(1^\lambda, (pk, ct)) : (pk, ct) \leftarrow \mathcal{D}_1]| \\ \neq 1. \end{gathered}$$

$$\Rightarrow (Gen, Enc, Dec) \text{ does not satisfy correctness,}$$

and it also means that Dec, in this case, must be a probabilistic algorithm, rather than a deterministic one; a deterministc Dec would satisfy security but fall to an unbounded $\mathcal{A}$.

Part B.

Given: $|ct| \leq Clog\lambda$, for some constant C.
Claim: (Gen, Enc, Dec) *does not* satistify CPA security.
Proof: This time, define $\mathcal{D}_b$ for *any* pair of messages m_0, m_1 chosen by some PPT $\mathcal{A}$, as

$$\mathcal{D}_b := \{(pk, Enc(pk, m_b)) : (pk, sk) \leftarrow Gen(1^\lambda)\}, \quad b \in \{0, 1\}.$$

We will show an $\mathcal{A}$, such that it can guess the message that goes with the ciphertext given to it with probability greater than $\frac{1}{2}$. Formally,

$$|\mathcal{P}[b = b'; b' \leftarrow \mathcal{A}(1^\lambda, (pk, ct)) : (pk, ct) \leftarrow \mathcal{D}_b]$$

$$\geq \frac{1}{2} + \epsilon(\lambda),$$

where $\epsilon(\lambda)$ is non-negligible. That is, $\epsilon(\lambda) = \frac{1}{q(\lambda)}$, where $q(\lambda)$ is some polynomial. We will show a PPT algorithm $\mathcal{A}$ such that

$$\mathcal{P}[\mathcal{A} \text{ successfully "wins" the game above}] = \frac{1}{2} + \frac{1}{q(\lambda)},$$

for some polynomial $q(\lambda)$. Here, $\mathcal{A}$ wins if it outputs a $b' = b$.
Take the set of all possible ciphertexts, given λ:

$$\mathcal{S} := \{Enc(pk, m) : m \leftarrow \{0, 1\}^*, |Enc(pk, m)| \leq Clog\lambda, pk \leftarrow Gen(1^\lambda)\}$$

We know that the size of $\mathcal{S}$ is the sum of the sets of all possible ciphertexts of lengths 1, 2, ..., λ. Thus,

$$|\mathcal{S}| = \sum_{i=0}^{Clog\lambda} |\{0, 1\}^i| = \sum_{i=0}^{Clog\lambda} 2^i = 2^{Clog\lambda + 1} - 1 = 2^C\lambda - 1,$$

since the right-most series above is geometric; we go on to design $\mathcal{A}$:

$$\mathcal{A}(1^\lambda, pk, ct) \rightarrow \begin{cases} 0 & \text{if } ct \leftarrow \mathcal{D}_0, \\ 1 & \text{if } ct \leftarrow \mathcal{D}_1, \\ x \xleftarrow{\$} \{0, 1\} & \text{otherwise.} \end{cases}$$

Define $ct_0 := Enc(pk, m_0)$ and $ct_1 := Enc(pk, m_1)$. ct is arbitrary, given to $\mathcal{A}$. Given $ct \in \{ct_0, ct_1\}$, we break this into 2 cases, to see when $\mathcal{A}$ wins (because if $ct \notin \{ct_0, ct_1\}$), we can resample (without replacement) our own random ct^* and set $ct = ct^*$ and run the same thing until $ct \in \{ct_0, ct_1\}$. We know eventually ct will be one of the two ciphertexts because of the amount of total possible ciphertexts.

- Case 1: $b = 0$.
  We want the probability that $\mathcal{A}$ wins, i.e. $\mathcal{A} \to 0$:
  $$\mathcal{P}[\mathcal{A} \text{ outputs } 0] = \mathcal{P}[\mathcal{A} \to 0 | ct = ct_0].$$
  By Law of Total Probability, we have $\mathcal{P}[\mathcal{A} \to 0 | ct = ct_0]$ is
  $$\begin{aligned}
  &\geq \mathcal{P}[\mathcal{A} \to 0]\mathcal{P}[ct = ct_0] + \mathcal{P}[\mathcal{A} \to 0]\mathcal{P}[ct \neq ct_0] \\
  &= 1 * \mathcal{P}[ct = ct_0] + \frac{1}{2} * \mathcal{P}[ct \neq ct_0] \\
  &= \frac{1}{2} * \mathcal{P}[ct = ct_0] + \frac{1}{2} * \mathcal{P}[ct = ct_0] + \frac{1}{2} * \mathcal{P}[ct \neq ct_0] \\
  &= \frac{1}{2}\mathcal{P}[ct = ct_0] + \frac{1}{2}(\mathcal{P}[ct = ct_0] + \mathcal{P}[ct \neq ct_0]) \\
  &= \frac{1}{2}\mathcal{P}[ct = ct_0] + \frac{1}{2}(1) \\
  &\geq \frac{1}{2}\mathcal{P}[ct = ct_0 \in \mathcal{S}] + \frac{1}{2} \\
  &\geq \frac{1}{2}\mathcal{P}[(ct \in \mathcal{S}) \wedge (ct_0 \in \mathcal{S})] + \frac{1}{2},
  \end{aligned}$$
  where ct and ct_0 are independently chosen because $\mathcal{A}$ only selects ct_0, and is given ct. From $|\mathcal{S}|$, we know that
  $$\frac{1}{2}\mathcal{P}[((ct \in \mathcal{S}) \wedge (ct_0 \in \mathcal{S})) : ct \perp ct_0] + \frac{1}{2} = \frac{1}{2}\frac{1}{(2^C\lambda - 1)^2} + \frac{1}{2}$$
  $$\Rightarrow \mathcal{P}[\mathcal{A} \text{ wins given it outputs } 0] = \frac{1}{2}\frac{1}{(2^C\lambda - 1)^2} + \frac{1}{2}$$

- Case 2: $b = 1$.
  Here, we want $\mathcal{P}[\mathcal{A} \to 1]$. By symmetry,
  $$\mathcal{P}[\mathcal{A} \text{ outputs } 1] = \frac{1}{2}\frac{1}{(2^C\lambda - 1)^2} + \frac{1}{2}.$$

Now,

$$\mathcal{P}[b = b' : b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathcal{A}(1^\lambda, pk, ct)]$$

can be written as

$$\frac{1}{2}\mathcal{P}[\mathcal{A} \to 0 | ct = ct_0] + \frac{1}{2}\mathcal{P}[\mathcal{A} \to 1 | ct = ct_1]$$

$$\geq \frac{1}{2} * 2 * (\frac{1}{2(2^C\lambda - 1)^2} + \frac{1}{2})$$

$$= \frac{1}{(2^C\lambda - 1)^2} + \frac{1}{2}.$$

Here, we can finally see that our $q(\lambda) = 2(2^C\lambda - 1)^2$, and thus $\epsilon(\lambda)$ is non-negligible

$$\Rightarrow (Gen, Enc, Dec) \text{ does not satisfy CPA security.}$$

Part C.

Given: Instead, suppose that Enc uses, at most, $log\lambda$ random bits.
Claim: (Gen, Enc, Dec) does not satisfy CPA security.
Proof: This problem can be directly reduced to problem 3, part b. That is, there is only $log\lambda$ bits of randomness used, and thus a polynomial reduction is possible to convert this problem A into a problem B, where the the probability of an adversary winning the type of game we described in part b (no need to formally define this for part c) is **at least** as high as the probability of some PPT $\mathcal{B}$ solving problem B, as its guesses in the ciphertext space will still be directly related to $Clog\lambda$.

Problem 4.

Given: Let $(Gen, Sign, Ver)$ be a many-time secure signature scheme. Consider $(Gen', Sign', Ver')$, constructed in the following way:

- $Gen'()$ is the same as $Gen()$
- $Sign'(sk, m)$ samples a fresh key pair $(sk^*, vk^*) \leftarrow Gen'(1^\lambda)$, and outputs $\mu \leftarrow (vk^*, Sign(sk, vk^*), Sign(sk^*, m))$.
- $Ver'(vk, m, \mu)$ parses μ as $\mu = (vk^*, \mu_1, \mu_2)$, and outputs VALID only if $Ver(vk, vk^*, \mu_1)$ outputs VALID and $Ver(vk^*, m, \mu_2)$ outputs VALID.

Assume the verification keys lie in the message space.
Claim: $(Gen', Sign', Ver')$ is many-time secure.
Proof: Suppose not. $\Rightarrow \exists$ PPT $\mathcal{A}$ that can break the signature scheme $(Gen', Sign', Ver')$ with a non-negligible probability $\epsilon(\lambda)$. That is, we want some PPT $\mathcal{A}$ that can get Ver' to output VALID when it shouldn't, with probability $\epsilon(\lambda)$. But if our $\mathcal{A}$ makes Ver' output a wrong output with probability $\epsilon(\lambda)$, then one of the following two cases occur: either this $\mathcal{A}$ can give us a μ_1' such that $Ver(vk, vk^*, \mu_1')$ outputs valid and $\mu_1' \neq \mu_1$, or $\mathcal{A}$ can give us a μ_2' such that $Ver(vk^*, m, \mu_2')$ outputs valid and $\mu_2' \neq \mu_2$.

- <u>Case 1:</u> In this case, a PPT $\mathcal{A}$ can get Ver to output VALID when it shouldn't given a verification key vk, and a message vk^*; but this implies that $(Gen, Sign, Ver)$ is not multi-time secure $\Rightarrow$ a contradiction. This case implies that $\mathcal{A}$ somehow recovers sk in polynomial time.
- <u>Case 2:</u> In this case, a PPT $\mathcal{A}$ can get Ver to output VALID when it shouldn't given a verification key vk^*, and a message m; but this implies that $(Gen, Sign, Ver)$ is not multi-time secure $\Rightarrow$ a contradiction. This case implies that $\mathcal{A}$ somehow recovers sk^* is polynomial time.

Both cases led to a contradiction

$$\Rightarrow (Gen', Sign', Ver') \text{ is many-time secure.}$$

Problem 5.

I don't know

Problem 6.

Given: An RSA signature scheme $(Gen, Sign, Ver)$.
Define a security game as follows:

- Given the security parameter 1^λ, $\mathcal{A}$ chooses an m.
- Challenger samples a $(sk, vk) \leftarrow Gen(1^\lambda)$ and sends back $\mu \leftarrow Sign(sk, m)$ to $\mathcal{A}$.
- $\mathcal{A}$ outputs a message pair (m', μ');
- $\mathcal{A}$ wins if $(Ver(vk, m', \mu') =$ VALID$)$ and $(m' \neq m$ or $\mu' \neq \mu)$.

Claim: No PPT $\mathcal{A}$ can win the above game with a non-negligible probability.
Proof: Suppose not; $\Rightarrow \exists$ a PPT $\mathcal{A}$ that can win the above game with a non-negligible probability $\epsilon(\lambda)$. There's two cases here: either $m' \neq m$ or $\mu' \neq \mu$, and $Ver(vk, m', \mu')$ outputs VALID.
By the union bound, the probability $\mathcal{A}$ succeeds in either of the two cases is less than or equal to the sum of the probabilities that it succeeds in each case.

- Case 1: $m' \neq m$ and $\mu' = \mu$.
  In this case, $\mathcal{A}$ is able to get $Ver(vk, m', \mu')$ to output valid on a different message. That is, $\mathcal{A}$ is looking for a *different* message m' to sign, that gives us the same signature $\mu' = \mu$, such that
  $$(\mu' = m^e \pmod N) \wedge (\mu' = (m')^e \pmod N) \wedge (m' \neq m).$$
  But Ver is a deterministic algorithm, and thus can only return valid on a given signature and public key for **one** given message, which comes from correctness of $(Gen, Sign, Ver)$. So, we have reached a contradiction, as it is impossible for $\mathcal{A}$ to do what we described above.

- Case 2: $m' = m$ and $\mu' \neq \mu$.
  In this case, $\mathcal{A}$ is able to get $Ver(vk, m', \mu')$ to output valid on a different signature (but the same message). This implies that $\mathcal{A}$ finds a $\mu' \neq \mu$ such that
  $$m = \mu^e \pmod N = \mu'^e \pmod N.$$
  This, however, reduces to $\mathcal{A}$ having to factor N or somehow recover the secret key d in polynomial time $\Rightarrow$ a contradiction.

Those are the only ways for $\mathcal{A}$ to succeed in this case, because it cannot run brute force to find d, and because the Unique Prime Factorization Theorem tells us that $\mathcal{A}$ won't randomly stumble upon a different private key d that it can use the get Ver to output valid.

We reach a contradiction in all cases, and thus can conclude that No PPT $\mathcal{A}$ can win the above game with a non-negligible probability, that is, given a signature on a message, not only is it hard to forge a signature on a different message, but it is hard to forge a different signature on the same message.

Problem 7.

Given: An interactive protocol, where the prover P is trying to prove to the verifier V that the graphs (G_0, G_1) are isomorphic:

- V sends $b_1, b_2 \xleftarrow{\$} \{0,1\}$ to P.
- P picks $b_1', b_2' \xleftarrow{\$} \{0,1\}$, and two random permutations σ_1 and σ_2 on the vertex set. P calculates $H_1 = \sigma_1(G_{b_1'})$ and $H_2 = \sigma_2(G_{b_2'})$, and sends H_1 and H_2 to V.
- V picks a random index $i \xleftarrow{\$} \{1,2\}$ and sends it to P.
- P sends a permutation π on the vertex set to V.
- V accepts *if and only if* $\pi(H_i) = G_{b_i}$.

Claim: The protocol described does not have soundness.
Proof: All we have to show, is a prover P, that can trick the verifier V into thinking two graphs G_0 and G_1 are isomorphic, when they really aren't. Consider a scenario with some prover P and two non-isomorphic graphs G_0 and G_1. We will show a P such that the verifier V always says two non-isomorphic graphs are, indeed isomorphic. Rephrasing the statement slightly, this is the same as showing that given our prover P, the following is true:

$$(V \text{ accepts } (G_0, G_1)) \Rightarrow (G_0 \text{ is not isomorphic to } G_1).$$

From the above statement, we now can assume that V accepts G_0 and G_1 as isomorphic graphs, which means $\pi(H_i) = G_{b_i}$, which means either $\pi(H_1) = G_{b_1}$ or $\pi(H_2) = G_{b_2}$. Consider the following prover P:

- the protocol above happens as described, until right before P sends a permutation π on the vertex set to V.
- here, prover P knows i, and thus knows which H_i to find a permutation for to obtain G_{b_i}.
- the prover P sends $\sigma_i^{-1}(H_i)$ back to the verifier V

It's important to note, that $\nexists$ a permutation that would take us from G_0 to G_1, or vice versa, which implies $\nexists$ a permutation that could take us from H_1 to $G_{b_2'}$, and likewise there's no permutation that would take us from H_2 to $G_{b_1'}$, *because* G_0 and G_1 are not isomorphic.

This means that, in this protocol, verifier V will actually only accept graphs that are *non-isomorphic* from our prover P.

$\Rightarrow$ the protocol described above does not have soundness.

Bonus Problem.

Part A.

The idea here is to construct g using f, since otherwise there's no way to prove that g is actually a one-way function. We can use the idea that's very similar to parts b and c below. Using f, we can construct a $g : \{0,1\}^\lambda \to \{0,1\}^3$, such that $g(x)$ is equal to the 3 least-significant binary digits of the output $f(x)$. The function g is one way, since the opposite would imply that f is not one way. And, g makes an RSA scheme insecure if used for hashing there, simply because we will be running RSA on a 3-bit output of g, and thus any PPT $\mathcal{A}$ would easily break such a scheme with good probability.

Part B.

Given: $H : \{0,1\}^m \to \{0,1\}^\lambda$ is a collision-resistant function, where $m(\lambda)$ is a polynomial function. We define H' as follows – pick three distinct inputs $x_1, x_2, x_3 \in \{0,1\}^m$, and set

$$H'(x) := \begin{cases} H(x)||0, & x \notin \{x_1, x_2, x_3\} \\ 3, & x = x_1 \\ 5, & x = x_2 \\ 15, & x = x_3, \end{cases}$$

where values $\{3, 5, 15\}$ correspond to their $(\lambda+1)$-bit representation.
Claim: H' is collision-resistant.
Proof: Suppose not. $\Rightarrow \exists$ a PPT $\mathcal{A}$ that can find $x, y \in \{0,1\}^m$, with $x \neq y$, s.t. $H'(x) = H'(y)$ with non-negligible probability $\epsilon(\lambda)$. Since x_1, x_2, x_3 are distinct inputs, the only way for out $\mathcal{A}$ to do this is when $x, y \notin \{x_1, x_2, x_3\}$. Thus, $\mathcal{A}$ can find x, y such that

$$H(x)||0 = H(y)||0,$$

$$\Rightarrow \mathcal{A} \text{ found } x \neq y \text{ s.t. } H(x) = H(y)$$

with probability $\epsilon(\lambda)$. This is a contradiction, showing that breaking collision-resistance of H' implies breaking collision-resistance of H,

$$\Rightarrow H' \text{ is collision-resistant.}$$

Part C.

Claim: $\exists$ a collision-resistant function H' such that if H' was used for hashing in the RSA signature scheme, it becomes insecure.

Proof: Consider the following H' – we pick 2^λ distinct inputs $x_1, x_2, ..., x_{2^\lambda} \in \{0,1\}^m$, and set

$$H'(x) := \begin{cases} (\lambda+1) - \text{bit representation of x}, & x = x_i, i \in [1, 2^\lambda] \\ H(x)||0, & \text{otherwise.} \end{cases}$$

That is, $H'(x)$ individually maps the first 2^λ inputs to their $(\lambda+1)$-bit binary representation, and uses $H(x)$ for the rest of the inputs. It's clear to see that, if this hash was used for an RSA encryption scheme, a PPT adversary $\mathcal{A}$ would be able to distinguish 2^λ encryptions, and thus be able to break an RSA encryption scheme $(Gen, Sign, Ver)$ with non-negligible probability. We could show the exact probability that this $\mathcal{A}$ can break the RSA scheme by correctly winning the sort of game described in problem 3; but it is enough that we know that m is polynomial in λ, and thus the probability of a ciphertext given to $\mathcal{A}$ being one of the ciphertexts $\mathcal{A}$ chose would be $\frac{1}{poly(\lambda)}$, and thus we have proven that this H', when used for hashing, makes an RSA scheme $(Gen, Sign, Ver)$ insecure.