Foundations of Cryptography

Solution 3

Ron Mokady

1. In class, we've seen the GGM PRF) where both the keys s and the inputs x were of length n. We would like to extend this construction so that for keys $s \in \{0,1\}^n$, we can apply the function for inputs of arbitrary length $x \in \{0,1\}^*$.

   (a) Consider applying the GGM function applied as is for inputs of arbitrary length. That is, for any $x \in \{0,1\}^*$, and letting ℓ be the length of x, and G a length doubling PRG, define

   $$F_s(x) := G_{x_\ell}(G_{x_{\ell-1}}(\cdots G_{x_2}(G_{x_1}(s))\cdots)) \ .$$

   We show that F is not a PRF:
   If F_s is a PRF By definition we know that for any n.u. PPT $A = \{A_n\}_{n\in\mathbb{N}}$ there exists a negligible μ such that for all $n \in \mathbb{N}$:

   $$\left|\Pr\left[A_n^{F_s(\cdot)} = 1 \;\middle|\; s \leftarrow \{0,1\}^n\right] - \Pr\left[A_n^{R(\cdot)} = 1 \;\middle|\; R \leftarrow \{0,1\}^{\{0,1\}^n}\right]\right| \leq \mu(n) \ .$$

   Let's define Adversary $\mathcal{B}$ such that query the string $'1'$ (length 1, answer is $F_s(1)$), then query the string $'11'$ (length 2, answer is $F_s(11)$) and return 1 iff $G_1(F_s(1)) = F_s(11)$. Thus,

   $$\Pr_{s\leftarrow\{0,1\}^n}\left[\mathcal{B}^{F_s(\cdot)} = 1\right] = \Pr_{s\leftarrow\{0,1\}^n}\left[G_1(F_s(1)) = F_s(11)\right] = \Pr_{s\leftarrow\{0,1\}^n}\left[G_1(G_1(s)) = G_1(G_1(s))\right] = 1$$

   $$\Pr_{s\leftarrow\{0,1\}^n}\left[\mathcal{B}^{F_s(\cdot)} = 0\right] = \Pr\left[G_1(U_n^{(1)}) = U_n^{(2)}\right] = 2^{-n}$$

   Hence,

   $$\left|\Pr\left[A_n^{F_s(\cdot)} = 1 \;\middle|\; s \leftarrow \{0,1\}^n\right] - \Pr\left[A_n^{R(\cdot)} = 1 \;\middle|\; R \leftarrow \{0,1\}^{\{0,1\}^n}\right]\right| = 1 - 2^{-n}$$

   Note that $\mathcal{B}$ is PPT since it only run G and F_s, so we get that F_s is not a PRF.

   (b) Consider the following variant of GGM. Let $G : \{0,1\}^n \to \{0,1\}^{3n}$ be a length-tripling PRG, and denote by $G_0(s), G_1(s), G_2(s)$ the first, second, and third blocks of n output bits, respectively. For any $x \in \{0,1\}^*$, and letting ℓ be the length of x, define

   $$F_s(x) := G_2(G_{x_\ell}(G_{x_{\ell-1}}(\cdots G_{x_2}(G_{x_1}(s))\cdots)) \ .$$

   We show that F is a PRF:

   The tree start from root s and each eternal node v has three children $G_0(v), G_1(v), G_2(v)$ so the two first are eternal nodes and the last is a leaf.
   We assume toward contradiction there is n.u. PPT adversary $\mathcal{A}$ such that distinguish between $F_s(U_n)$ and U_n with non negligible probability ε.
   Note that the definition of PRF regards only n.u. PPT Adversaries. For some n.u. PPT adversary $A = \{A_n\}_{n\in\mathbb{N}}$, since it's polynomial-time, we know exists polynomial d such that on input 1^n, A makes only queris of length at most d(n)-1.

We now consider $d(n)+1$ hybrid experiments $H_0, \ldots, H_{d(n)}$ where we gradually change the labels. In H_i, all the labels $L_{x_1 \ldots x_i}$ for nodes at level i are sampled independently at random. Then, the rest of the labels down the tree are chosen according to the same recursive rule as before. In each of these hybrid experiments, the adversary's queries are answered according to the leaf labels.
Then, there exists an $i \in [n]$ such that A distinguishes H_{i-1} from H_i with advantage $\varepsilon/d(n)$.
We now construct the distinguisher B. One difference from the proof was shown in class is that know A might query about levels $j < i$. In this case we answer randomally and remember our answer for future queries.
For intuition, note that by definition $F_s(x)$, an answer to a query is always a leaf, thus we can't calculate answer to a query by other answers without "break" G.

Algorithm $B(Y^{(1)}, \ldots, Y^{(q)})$**:**

- Initialize $\mathcal{L} = \emptyset$ (the set of i-level labels encountered so far).
- Initialize $\mathcal{L}^* = \emptyset$ (for every $j < i$ the set contain the j-level labels encountered so far).
- Initialize a counter $c = 0$ (the number of samples used so far from the list $Y^{(1)}, \ldots, Y^{(q)}$).
- Emulate A, and when it makes a query x act as follows:
  i. If $|x| < i-1$:
     - If $\mathcal{L}^*$ contain a pair (x, s), return s.
     - else, choose randomally $s \leftarrow \{0,1\}^n$, add (x, s) to $\mathcal{L}^*$ and return s.
  ii. If $|x| = i-1$ and $\mathcal{L}$ contain a pair $(x_1 \ldots x_{i-1}, Y)$, return Y.
  iii. If $|x| = i-1$ and $\mathcal{L}$ has no such element:
     - Use the next sample $Y^{(c+1)} = Y_0^{(c+1)} Y_1^{(c+1)} Y_2^{(c+1)}$ to add $(x_1 \ldots x_{i-1}0, Y_0^{(c+1)}), (x_1 \ldots x_{i-1}1, Y_1^{(c+1)}), (x_1 \ldots x_{i-1}, Y_2^{(c+1)})$ to the list $\mathcal{L}$.
     - Increase the counter c by one.
     - Return $Y_2^{(c+1)}$.
  iv. If $|x| > i-1$ and $\mathcal{L}$ contains a pair $(x_1 \ldots x_i, L_{x_1 \ldots x_i})$, compute the labels down the path from $x_1 \ldots x_i$ to $x_1 \ldots x_n$ according to the recursive rule, and return the corresponding leaf.
  v. If $|x| > i-1$ and $\mathcal{L}$ has no such element:
     - Use the next sample $Y^{(c+1)} = Y_0^{(c+1)} Y_1^{(c+1)} Y_2^{(c+1)}$ to add $(x_1 \ldots x_{i-1}0, Y_0^{(c+1)}), (x_1 \ldots x_{i-1}1, Y_1^{(c+1)}), (x_1 \ldots x_{i-1}, Y_2^{(c+1)})$ to the list $\mathcal{L}$.
     - Increase the counter c by one.
     - Return to step (2).
- At the end output whatever A outputs.

It is left to see that if the samples $Y^{(1)}, \ldots, Y^{(q)}$ are pseudorandom, then A's emulated view is exactly as in H_{i-1}, whereas if they are random, then it's exactly as in H_i. Thus, B manages to distinguish the two cases with advantage $\varepsilon/d(n)$ in contradiciton to the definition of PRG. This means that $\epsilon(n) = n^{-\omega(1)}$ as required.

2. Let $(Auth, Ver)$ be a MAC, and assume $Auth$ is deterministic (i.e., it does not toss any coins on its own) and that for secret key of size n it produces tags of size n.

   (a) Consider the function $G : \{0,1\}^{2n} \to \{0,1\}^{3n}$ defined as:

   $$G(sk, r) = r, \langle Auth_{sk}(1), r\rangle, \langle Auth_{sk}(11), r\rangle, \ldots, \langle Auth_{sk}(1^{2n}), r\rangle \ ,$$

   where $\langle \cdot, \cdot \rangle$ denotes inner product modulo 2. We show that G is a PRG.

Let's assume toward contradiction that G is not a PRG, then exist $\mathcal{A}'$ and polynomial p' such that

$$|\Pr[\mathcal{A}'(G(U_{2n}))=1]-\Pr[\mathcal{A}'(U_{3n})=1]|\geq\frac{1}{p'(n)}$$

Now let's define the following hybrids for $0\leq i\leq 2n$, $r\leftarrow\{0,1\}^n, sk\leftarrow\{0,1\}^n$:

$$H_{i,n}=r,\langle Auth_{sk}(1),r\rangle,\ldots,\langle Auth_{sk}(1^i),r\rangle,U_{2n-i}$$

Note that $H_{0,n}=U_3n$ nad if $r\leftarrow\{0,1\}^n, sk\leftarrow\{0,1\}^n$ $H_{2n,n}=G(sk,r)$. By traingle inequality we get that exist i such that

$$|\Pr[\mathcal{A}'(H_{i,n})=1]-\Pr[\mathcal{A}'(H_{i+1,n})=1]|\geq\frac{1}{2n\cdot p'(n)}$$

Now we define the distribution ensembles $X=\{X_n\}_{n\in\mathbb{N}}$, $Y=\{Y_n\}_{n\in\mathbb{N}}$:

$$X=r,\langle Auth_{sk}(1),r\rangle,\ldots,\langle Auth_{sk}(1^i),r\rangle$$

$$B=\langle Auth_{sk}(1^{i+1}),r\rangle$$

We want to create adversary that distinguish between X,B and X,U_1. Let's define adversary $\mathcal{A}$ that get (x,b) as input, randomize $s\leftarrow\{0,1\}^{2n-i-1}$, run $\mathcal{A}'$ on the concatenation of x,b,s and output the same.
Note that XBU_{2n-i-1} distributes exactly like $H_{i+1,n}$ and XU_1U_{2n-i-1} distributes exactly like $H_{i,n}$, so by definition of $\mathcal{A}$ we get that:

$$\begin{aligned}&|\Pr[\mathcal{A}(X,B)=1]-\Pr[\mathcal{A}(X,U_1)=1)]|\\&=|\Pr[\mathcal{A}'(XBU_{2n-i-1})=1]-\Pr[\mathcal{A}'(XU_1U_{2n-i-1})=1]|\\&=|\Pr[\mathcal{A}'(H_{i,n})=1]-\Pr[\mathcal{A}'(H_{i+1,n})=1]|\geq\frac{1}{2n\cdot p'(n)}\end{aligned}$$

Now we use the fact that for any distribution X and jointly distributed bit B, if A distinguishes X,B from X,U_1 with advantage ε, there exists a predictor P that given X predicts B with probability $1/2+\varepsilon/2$ (where P runs in time polynomially related to that of A).
Hence, exist predictor $\mathcal{P}$ that given X predicts $\langle Auth_{sk}(1^{i+1}),r\rangle$ with probability $1/2+\frac{1}{4n\cdot p'(n)}$.
We now define $\mathcal{P}^*_{sk}$ that given input $r\in\{0,1\}^n$, run $\mathcal{P}$ on the input $(r,\langle Auth_{sk}(1),r\rangle,\ldots,\langle Auth_{sk}(1^i),r\rangle)$ and output tha same. Hence, $\mathcal{P}^*_{sk}$ given r predicts $\langle Auth_{sk}(1^{i+1}),r\rangle$ with probability $1/2+\frac{1}{4n\cdot p'(n)}$.
According to claim we saw in class there is a fraction of $\frac{1}{2n\cdot p'(n)}$ of sk possible values such that $\mathcal{P}^*_{sk}$ given r predicts $\langle Auth_{sk}(1^{i+1}),r\rangle$ with probability $1/2+\frac{1}{2n\cdot p'(n)}$ we define set of those value as S.
According to GL for $sk\in S$:

$$\text{if }\Pr_r\left[\tilde{L}_x(r)=\langle x,r\rangle\mod 2\right]\geq 1/2+\delta\ ,\text{ then }\Pr\left[GL^{\tilde{L}_x(\cdot)}_{n,\delta}=x\right]\geq\Omega(\delta^2/n)\ .$$

Since $Auth$ is deterministic we can use GL, Thus for $sk\in S$:

$$\Pr\left[GL^{\mathcal{P}^*_{sk}(\cdot)}_n=Auth_{sk}(1^{i+1})\right]\geq\Omega(\frac{1}{4n^3\cdot p'^2(n)})\ .$$

We define new adversary $\mathcal{B}$ that uses $GL^{\mathcal{P}^*_{sk}(\cdot)}_n$ to cumpute $Auth_{sk}(1^{i+1})$: given r from $GL^{\mathcal{P}^*_{sk}(\cdot)}_n$ $\mathcal{B}$ use $Auth_{sk}$ to compute $(r,\langle Auth_{sk}(1),r\rangle,\ldots,\langle Auth_{sk}(1^i),r\rangle)$ and pass it to our predicator $\mathcal{P}$

that output prediction p, then pass the prediction p to the $GL_n^{\mathcal{P}^*_{sk}(\cdot)}$ algorithm which then output $Auth_{sk}(1^i)$, $\mathcal{B}$ output $(1^i, Auth_{sk}(1^i))$. Hence,

$$\begin{aligned}
&\Pr\left[(m^*,t^*) \leftarrow \mathcal{B}^{Auth^*_{sk}(\cdot)} \;\middle|\; m^* \notin Q, Ver^*_{sk}(m^*,t^*)=1\right] \\
&\geq \Pr\left[(1^i, Auth_{sk}(1^i)) \leftarrow \mathcal{B}^{Auth^*_{sk}(\cdot)} \;\middle|\; Auth_{sk}(1^i) \notin Q, Ver^*_{sk}(1^i, Auth_{sk}(1^i))=1 \text{ and } sk \in S\right] \Pr\left[sk \in S\right] \\
&= \Pr\left[GL_n^{\mathcal{P}^*_{sk}(\cdot)} = Auth_{sk}(1^{i+1})\right] \frac{1}{2n \cdot p'(n)} \geq \Omega(\frac{1}{4n^3 \cdot p'^2(n)}) \cdot \frac{1}{2n \cdot p'(n)}
\end{aligned}$$

In contradiction to the fact that $(Auth, Ver)$ is a MAC. Thus, G is a PRG.

(b) (**Bonus**) Consider the function $G : \{0,1\}^{2n} \to \{0,1\}^{3n+1}$ defined as:

$$G(sk, r) = r, \langle Auth_{sk}(r), r\rangle, \langle Auth_{sk}(1), r\rangle, \langle Auth_{sk}(11), r\rangle, \ldots, \langle Auth_{sk}(1^{2n}), r\rangle \ .$$

we prove that this function is not a PRG:
First, given MAC $(Auth, Ver)$ we will construct new MAC $(Auth^*, Ver^*)$ in the following way:
Given binary message m define $J(m)$ as the index of the first bit which is 1 in m. Now let's define function F which receive input s and index j, and output the input after fliping (1 to 0 or 0 to 1) the bit in the index j.
So $F(s, J(t))$ output s after fliping the first bit which is 1 in t. We conclude that if $\langle s,t\rangle = 1$ then $\langle F(s, J(t)), t\rangle = 0$. And if $\langle s,t\rangle = 0$ then $\langle F(s, J(t)), t\rangle = 1$.
Now we define

$$Auth^*_{sk}(r) = \begin{cases} Auth_{sk}(r), & \text{if } \langle Auth_{sk}(r), r\rangle = 1 \\ F(Auth_{sk}(r), J(r)), & \text{otherwise} \end{cases}$$

$$Ver^*_{sk}(m,t) = \begin{cases} 1, & \text{if } Ver_{sk}(m,t) = 1 \text{ or } Ver_{sk}(m, F(t, J(m))) = 1 \\ 0, & \text{otherwise} \end{cases}$$

By this definition we get correctness:

$$\begin{aligned}
&\Pr[Ver^*_{sk}(r, Auth^*_{sk}(r)) = 1] \\
&= \Pr\left[Ver^*_{sk}(r, Auth^*_{sk}(r)) = 1 \mid \langle Auth_{sk}(r), r\rangle = 1\right] \Pr[\langle Auth_{sk}(r), r\rangle = 1] \\
&+ \Pr\left[Ver^*_{sk}(r, Auth^*_{sk}(r)) = 1 \mid \langle Auth_{sk}(r), r\rangle = 0\right] \Pr[\langle Auth_{sk}(r), r\rangle = 0] \\
&= \Pr\left[Ver^*_{sk}(r, Auth_{sk}(r)) = 1 \mid \langle Auth_{sk}(r), r\rangle = 1\right] \Pr[\langle Auth_{sk}(r), r\rangle = 1] \\
&+ \Pr\left[Ver^*_{sk}(r, F(Auth_{sk}(r), J(r))) = 1 \mid \langle Auth_{sk}(r), r\rangle = 0\right] \Pr[\langle Auth_{sk}(r), r\rangle = 0] \\
&= \Pr[\langle Auth_{sk}(r), r\rangle = 1] + \Pr[\langle Auth_{sk}(r), r\rangle = 0] = 1
\end{aligned}$$

We want to prove unforgeability against chosen plaintext atteck, hence we assume toward contradiction that $(Auth^*, Ver^*)$ is not unforgeability against chosen plaintext atteck, so exist oracle $\mathcal{A}'$ and polynomial p' such that

$$\Pr\left[(m^*,t^*) \leftarrow \mathcal{A}'^{Auth^*_{sk}(\cdot)} \;\middle|\; m^* \notin Q, Ver^*_{sk}(m^*,t^*)=1\right] \geq \frac{1}{p'(n)} \ ,$$

Let's define oracle $\mathcal{A}$:

- Run $\mathcal{A}'$ that output (m^*, t^*).
- randomize $b \leftarrow \{0,1\}^n$.
- If $b = 0$ output (m^*, t^*), else output $(m^*, F(t^*, J(m^*))$.

Note that if $b = 0$ and $\langle Auth_{sk}(r), r\rangle = 1$, $\mathcal{A}$ output a valid (m^*, t^*). Similarily if $b = 1$ and $\langle Auth_{sk}(r), r\rangle = 0$, $\mathcal{A}$ output a valid (m^*, t^*). Thus we get that:

$$\Pr\left[(m^*, t^*) \leftarrow \mathcal{A}'^{Auth_{sk}(\cdot)} \;\middle|\; m^* \notin Q, Ver_{sk}(m^*, t^*) = 1\right] \geq \frac{1}{2 \cdot p'(n)} \ ,$$

In contradiction to the fact that $(Auth, Ver)$ is a MAC, hence $(Auth^*, Ver^*)$ is a MAC.

Now it's left to notice that by defintion of $(Auth^*, Ver^*)$ always $\langle Auth^*_{sk}(r), r\rangle = 1$, hence if G uses $(Auth^*, Ver^*)$ as MAC we can build adversary $\mathcal{B}$ that distinguish G: $\mathcal{B}$ output 1 iff the $(n+1)$th bit of the input is 1. So if $\mathcal{B}$ gets as input the output of G it is guaranteed that $\langle Auth^*_{sk}(r), r\rangle = 1$, therfore $\mathcal{B}$ output 1. Thus,

$$|\Pr[\mathcal{B}(G(U_{2n})) = 1] - \Pr[\mathcal{B}(U_{3n+1}) = 1]| = 1 - \frac{1}{2} = \frac{1}{2}$$

Overall we got that G is not a PRG as required.

3. Let H be a collision-resistant hash function that for a key $hk \in \{0,1\}^n$ maps $\{0,1\}^{2n}$ to $\{0,1\}^n$. We would like to use H to construct a new collision-resistant H^* for inputs of arbitrary length. We first focus on inputs whose length is a power of two, 2^ℓ for some ℓ. We consider the following construction (known as *Merkle's tree hash*).

   Given a key hk for H, and an input x of length 2^ℓ, consider the following labeling $\{L_v\}_v$ of the binary tree of depth ℓ whose nodes are represented by $\left\{v \in \{0,1\}^i, i \in \{0,\ldots,\ell\}\right\}$:

   - For every leaf $v \in \{0,1\}^\ell$, $L_v := x_v 0^{2n-1}$ (the vth bit of x padded).
   - For every intermediate node, $L_v = H_{hk}(L_{v0} L_{v_1})$ (the hash of its sons).

   The hash $H^*_{hk}(x)$ is then set to be the root label L_ε.

   (a) We show that H^* is also collision resistant (for inputs whose length is a power of two):
   We know that H is collision-resistent, hence for any n.u. PPT $A = \{A_n\}_{n\in\mathbb{N}}$ there exists a negligible μ such that for all $n \in \mathbb{N}$,

   $$\Pr\left[\begin{array}{c|c} H_{hk}(x) = H_{hk}(x') & hk \leftarrow \{0,1\}^n \\ x \neq x' & (x, x') \leftarrow A_n(hk) \end{array}\right] \leq \mu(n) \ .$$

   Let's assume toward contradiction that H^*_{hk} is not collision resistent so there is n.u. PPT $\mathcal{A}'$ and polynomial $p'(n)$ such that

   $$\Pr\left[\begin{array}{c|c} H^*_{hk}(x) = H^*_{hk}(x') & hk \leftarrow \{0,1\}^n \\ x \neq x' & (x, x') \leftarrow \mathcal{A}'(hk) \end{array}\right] \geq \frac{1}{p'(n)} \ .$$

   Note that given x such that $|x| = 2^\ell$, the number of nodes in the merkle tree of $H^*_{hk}(x)$ is $2^\ell + 2^{\ell-1} + \ldots + 1 < 2 \cdot 2^\ell$. Hence, we can compute the merkle tree of $H^*_{hk}(x)$ in polynomial time.
   Let's denote L^x_v the label of the node v in the merkle tree of $H^*_{hk}(x)$ and denote $v0$ (respectedly $v1$) the left (respectedly right) child of v. The root node denote as ε.
   Now we prove that given $x \neq x'$, if $H^*_{hk}(x) = H^*_{hk}(x')$, exists vertex u so $L^x_u = L^{x'}_u$ and $L^x_{u0} L^x_{u1} \neq L^{x'}_{u0} L^{x'}_{u1}$.
   Given $x \neq x'$ and $|x| = |x'| = 2^\ell$, we prove it by induction of ℓ:
   Base case: for $\ell = 1$, if $H^*_{hk}(x) = H^*_{hk}(x')$ then $L^{x'}_\varepsilon = L^x_\varepsilon$, and by definition of the H^* tree we get that the leaves of the merkle trees creating x and x' are different because $x \neq x'$, so $L^x_{\varepsilon 0} L^x_{\varepsilon 1} \neq L^{x'}_{\varepsilon 0} L^{x'}_{\varepsilon 1}$ as required.
   Let's assume it holds for $\ell = \ell'$ and prove it for $\ell = \ell' + 1$: Given $x \neq x'$ and $|x| = |x'| = 2^{\ell+1}$

if $H^*_{hk}(x) = H^*_{hk}(x')$ then $L^{x'}_{\varepsilon} = L^x_{\varepsilon}$. let's split to cases, in the first case $L^x_{\varepsilon 0}L^x_{\varepsilon 1} \neq L^{x'}_{\varepsilon 0}L^{x'}_{\varepsilon 1}$ as required. In the second case we get $L^x_{\varepsilon 0}L^x_{\varepsilon 1} = L^{x'}_{\varepsilon 0}L^{x'}_{\varepsilon 1}$. Let's denote x_0 as the 2^ℓ most significant bits of x and by x_1 the rest of 2^ℓ bits of x . Because $x \neq x'$ we know that $x_0 \neq x'_0$ or $x_1 \neq x'_1$, assume W.L.O.G that $x_0 \neq x'_0$. Note that by definitio of the H^* tree, the tree whos root is $L^x_{\varepsilon 0}$ is the merkle tree created by x_0 (the same holds for x') and $L^x_{\varepsilon 0} = H^*_{hk}(x_0)$, thus $H^*_{hk}(x'_0) = H^*_{hk}(x_0)$, and by the induction hypothesis we know exists vertex u so $L^x_u = L^{x'}_u$ and $L^x_{u0}L^x_{u1} \neq L^{x'}_{u0}L^{x'}_{u1}$ as required.
We just proved that if $H^*_{hk}(x) = H^*_{hk}(x')$, exists vertex u so $L^x_u = L^{x'}_u$ and $L^x_{u0}L^x_{u1} \neq L^{x'}_{u0}L^{x'}_{u1}$. so if $H^*_{hk}(x) = H^*_{hk}(x')$ exists u so $H_{hk}(L^x_{u0}L^x_{u1}) = H_{hk}(L^{x'}_{u0}L^{x'}_{u1})$ and $L^x_{u0}L^x_{u1} \neq L^{x'}_{u0}L^{x'}_{u1}$, means that if $H^*_{hk}(x) = H^*_{hk}(x')$ the correspond trees contain $y \neq y'$ so $H_{hk}(y) = H_{hk}(y')$.

Let's define adversary $\mathcal{A}$ so that on input hk: Run $\mathcal{A}'(hk)$, and from the output x, x' build the merkle trees corresponding to H^*_{hk}. Then search the trees for vertex u so $L^x_u = L^{x'}_u$ and $L^x_{u0}L^x_{u1} \neq L^{x'}_{u0}L^{x'}_{u1}$ and output $(L^x_{u0}L^x_{u1}, L^{x'}_{u0}L^{x'}_{u1})$.
The correctness follows as we just proved that exists such u, the tree building can be done in polynomial time and we showed that $H_{hk}(L^x_{u0}L^x_{u1}) = H_{hk}(L^{x'}_{u0}L^{x'}_{u1})$ as required.

Hence, exist adversary such

$$\Pr\left[\begin{array}{c|c} H_{hk}(x) = H_{hk}(x') & hk \leftarrow \{0,1\}^n \\ x \neq x' & (x,x') \leftarrow \mathcal{A}(hk) \end{array}\right] \geq \frac{1}{p'(n)} \ .$$

In contradiction to the fact that H is collision resistent. Hence, H^*_{hk} is collision resistent as required.

(b) We explain how to extend H^* to inputs that are not a power of two (without full proof):
Given x of length $n(x)$ exist $\ell(x)$ s.t. $2^{\ell(x)-1} \leq |x| \leq 2^{\ell(x)}$. If we define the binary representation of $|x|$ as $Bin(|x|)$ and $|Bin(|x|)| = k(x)$ then exist $\ell^*(x)$ s.t. $2^{\ell^*(x)-1} \leq k(x) \leq 2^{\ell^*(x)}$. We now define our new hash function that can get input of any length as:

$$H^{new}_{hk}(x) = H_{hk}(H^*_{hk}(x0^{2^{\ell(x)}-n(x)})H^*_{hk}(Bin(|x|)0^{2^{\ell^*(x)}-k(x)}))$$

In another words, we padd x to the length of power of two and perform H^*, and padd $|x|$ to the length of power of two and perform H^*, then we perform H on the two n-bit strings we got.

Proof sketch: Assume twoard negation that we can find $x \neq x'$ such $H^{new}_{hk}(x) = H^{new}_{hk}(x')$. First case is that $H^*_{hk}(x0^{2^{\ell(x)}-n(x)})H^*_{hk}(Bin(|x|)0^{2^{\ell^*(x)}-k(x)}) \neq H^*_{hk}(x'0^{2^{\ell(x')}-n(x')})H^*_{hk}(Bin(|x'|)0^{2^{\ell^*(x')}-k(x')})$ so we got $y \neq y'$ so $H_{hk}(y) = H_{hk}(y')$ in contradiction to the fact that H_{hk} is collision resistent. In Second case $H^*_{hk}(x0^{2^{\ell(x)}-n(x)})H^*_{hk}(Bin(|x|)0^{2^{\ell^*(x)}-k(x)}) = H^*_{hk}(x'0^{2^{\ell(x')}-n(x')})H^*_{hk}(Bin(|x'|)0^{2^{\ell^*(x')}-k(x')})$. If $Bin(|x|) \neq Bin(|x'|)$ then $Bin(|x|)0^{2^{\ell^*(x)}-k(x)} \neq Bin(|x'|)0^{2^{\ell^*(x')}-k(x')}$, and we got $y \neq y'$ so $H^*_{hk}(y) = H^*_{hk}(y')$ in contradiction to the last section. Else, $Bin(|x|) = Bin(|x'|)$, thus $|x| = |x'|$, hence $\ell(x) = \ell(x')$ so we got $x'0^{2^{\ell(x')}-n(x)} \neq x0^{2^{\ell(x)}-n(x')}$ but $H^*_{hk}(x'0^{2^{\ell(x')}-n(x)}) = H^*_{hk}(x'0^{2^{\ell(x)}-n(x')})$ in contradiction to the last section.