E0 235 : Cryptography	**Question Set**
Tutorial 3	
Instructor: Arpita Patra	*Feb 21, 2016*

Question 1

Let f, g be length-preserving one-way functions (so, e.g., $|f(x)| = |x|$). For each of the following functions f', decide whether it is necessarily a one-way function (for arbitrary f, g) or not. If it is, prove it. If not, show a counterexample.

(a) $f'(x) \stackrel{def}{=} f(x) \bigoplus g(x)$.

(b) $f'(x) \stackrel{def}{=} f(x)||g(x)$.

(c) $f'(x_1||x_2) \stackrel{def}{=} f(x_1)||g(x_2)$.

Question 2

Let f be a length-preserving one-way function. Let $bit(i, x) \stackrel{def}{=} x_i$, the i^{th} bit of x (defined for $1 \leq i \leq |x|$).

(a) Prove that the function f' defined by

$$f'(x) = f(x)||bit(1, x)||1$$

is one-way, but that the predicate $bit(1, \cdot) : \{0, 1\}^* \to \{0, 1\}$ is not hard-core for f'.

(b) Construct a function g that is one-way, but such that no bit of the input is hard-core.

Question 3

Prove that if there exist one-way functions, then there exists a one-way function f such that for every n, $f(0^n) = 0^n$. Provide a full (formal) proof of your answer.

Question 4

Show a CPA-secure private-key encryption scheme but is not CCA-secure.

Question 5

Show that if a one-to-one function has a hard-core predicate, then it is one-way.

Answers

1. (a) This f' is not (in general) a one-way function. To see this, take $f = g$ (i.e., set them to be the same function). Then f' maps all points to the all-0 string, and is certainly not one-way.

   (b) This f' is not (in general) a one-way function. For example, let g be a one-way function and define f as follows:
$$f(x_1 \| x_2) = g(x_2) \| 0^n,$$
   where $|x_1| = |x_2| = n$. It is not hard to see that f is one-way (a proof is left as an exercise). On the other hand, f' as defined in the problem maps all inputs to the constant value $g(0^n)\|0^n$, and so is not one-way.

   Interestingly, if f is a one-way *permutation* then f' must be one-way. A proof of this is also left as an exercise.

   (c) This f' **is** one-way. In fact, this holds even if only f is one-way (regardless of g, as long as g is efficiently-computable). To see this, fix a PPT adversary $\mathcal{A}'$ and let
$$\epsilon(n) \stackrel{\text{def}}{=} \Pr[\mathcal{A}'(f'(x)) \text{ outputs an inverse of } f'(x)],$$
   where the probability is taken over uniform choice of x and the random coins of $\mathcal{A}'$. Consider the following PPT adversary $\mathcal{A}$: given input y_1 (which is equal to $f(x_1)$ for randomly-chosen x_1), choose random x_2, compute $y_2 := g(x_2)$, and run $\mathcal{A}'(y_1 \| y_2)$. Then output the first half of the string output by $\mathcal{A}'$. It is not hard to see that (1) the input $y_1 \| y_2$ given to $\mathcal{A}'$ is distributed identically to $f'(x_1 \| x_2)$ for randomly-chosen x_1, x_2. This implies that $\mathcal{A}'$ inverts its input with probability $\epsilon(n)$. Furthermore, (2) whenever $\mathcal{A}'$ successfully inverts its input, $\mathcal{A}$ successfully inverts its own input. We conclude that $\mathcal{A}$ outputs an inverse of y_1 with probability at least $\epsilon(n)$, showing that ϵ must be negligible.

2. (a) It is immediate that $\mathsf{bit}(1, \cdot)$ is not hard-core for the given function f', so we just prove that f' is one-way. Fix some PPT adversary $\mathcal{A}'$ and let
$$\epsilon(n) \stackrel{\text{def}}{=} \Pr[\mathcal{A}'(f'(x)) \text{ outputs an inverse of } f'(x)].$$
   Construct the following adversary $\mathcal{A}$:

   > Given input y (which is equal to $f(x)$ for random x), choose a random bit b and run $\mathcal{A}'(y\|b\|1)$ to get x. Output x.

   To analyze the behavior of $\mathcal{A}$, note that $b = \mathsf{bit}(1, x)$ with probability at least $1/2$. (It can occur with higher probability if f is not one-to-one.) Furthermore, if $\mathcal{A}'$ outputs an inverse of $y\|b\|1$ then $\mathcal{A}$ correctly inverts its given input y. We conclude that $\mathcal{A}$ outputs an inverse of y with probability at least $\epsilon(n)/2$, and so ϵ must be negligible.

   (b) One possibility is to define $f'(x, i) = f(x) \, \| \, \mathsf{bit}(i, x) \, \| \, i$. Any bit of the input can be guessed with probability at least $1/2 + O(1/n)$ (where $|x| = n$), but it is possible to prove (as in part (a)) that f' is still one-way.

Answer 3

Exercise 2: Prove that if there exist one-way functions, then there exists a one-way function f such that for every n, $f(0^n) = 0^n$. Provide a full (formal) proof of your answer.

Solution 2: I provide a painfully detailed proof.

Let f be one-way function (that exists by the assumption) and define $g(x) = f(x)$ for every $x \neq 0^{|x|}$ and $g(0^n) = 0^n$ for every n. Clearly, g fulfills the requirements. It remains to prove that it is one-way. First, g is efficiently computable. Second, assume by contradiction that there exists a PPT algorithm A and a polynomial $p(\cdot)$ such that for infinitely many n's, algorithm A inverts g with probability at least $1/p(n)$. We begin by analyzing the probability that A succeeds in inverting g on non-zero inputs:

$$\begin{aligned}
\Pr\left[A(g(U_n)) \in g^{-1}(g(U_n))\right] &= \Pr\left[A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n\right] \cdot \Pr\left[U_n \neq 0^n\right] \\
&\quad + \Pr\left[A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n = 0^n\right] \cdot \Pr\left[U_n = 0^n\right] \\
&\leq \Pr\left[A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n\right] + \Pr\left[U_n = 0^n\right] \\
&= \Pr\left[A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n\right] + \frac{1}{2^n}
\end{aligned}$$

Therefore, for infinitely many n's we have that:

$$\Pr\left[A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n\right] \geq \frac{1}{p(n)} - \frac{1}{2^n}$$

We now construct B that inverts f as follows. Upon receiving an input y, B invokes A and returns whatever A outputs. We analyze B's success:

$$\begin{aligned}
\Pr\left[B(f(U_n)) \in f^{-1}(f(U_n))\right] &= \Pr\left[B(f(U_n)) \in f^{-1}(f(U_n)) \mid U_n \neq 0^n\right] \cdot \Pr\left[U_n \neq 0^n\right] \\
&\quad + \Pr\left[B(f(U_n)) \in f^{-1}(f(U_n)) \mid U_n = 0^n\right] \cdot \Pr\left[U_n = 0^n\right] \\
&\geq \Pr\left[B(f(U_n)) \in f^{-1}(f(U_n)) \mid U_n \neq 0^n\right] \cdot \Pr\left[U_n \neq 0^n\right] \\
&= \Pr\left[A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n\right] \cdot \left(1 - \frac{1}{2^n}\right) \\
&\geq \left(\frac{1}{p(n)} - \frac{1}{2^n}\right) \cdot \left(1 - \frac{1}{2^n}\right) > \frac{1}{2p(n)}
\end{aligned}$$

We conclude that for infinitely many n's, algorithm B inverts f with probability greater than $1/2p(n)$, in contradiction to the one-wayness of f.

Answer 4

Example of CPA secure system that's not CCA secure

$$Enc_k(m) = \langle r, F_k(r) \oplus m \rangle, where\ |m| = n, r \leftarrow U_n\ and\ F_k\ is\ a\ PRF$$

A CCA attack

Adversary sends $m_0 = 0^n$ and $m_1 = 1^n$ to the encryption oracle.

He gets back $Enc_k(m_b) = \langle r, F_k(r) \oplus m_b \rangle, b \leftarrow u$

- $Enc_k(m_b) = \begin{cases} Enc_k(m_0) = \langle r_0, F_k(r_0) \rangle, 50\%\ of\ the\ time \\ Enc_k(m_1) = \langle r_1, \overline{F_k(r_1)} \rangle, 50\%\ of\ the\ time \end{cases}$

Since the decryption oracle will not accept the encryption of the challenge messages (m_0 or m_1) as input, the attacker will do the next best thing. He can flip the first bit in the encryption of m_b.

The decryption oracle will gladly decrypt the new ciphertext $\langle r,\ c \oplus 10^{n-1} \rangle$.

The adversary can now tell from what he gets back (either 01^{n-1} or 10^{n-1}) whether $m_0 = 0^n$ or $m_1 = 1^n$ was encrypted.

Answer 5

Let f be a one-to-one function and b a hard-core predicate of f.

Intuition: When the function is one-to-one, all the information about the preimage x is found in $f(x)$. Therefore, it can only be hard to compute $b(x)$ if f cannot be inverted.

Proof: For contradiction, assume, that f is not one-way. There exists a PPT algorithm A and a non-negligible function ε such that A inverts $f(x)$ with probability at least $\varepsilon(n)$ (where the probability is taken over random choice of $x \leftarrow \{0,1\}^n$. We will construct an algorithm B that breaks the predicate b with non-negligible probability. B does the following

1. on input $f(x)$ it runs algorithm A on $f(x)$,
2. it takes the output x' of A and computes $b(x')$,
3. it outputs $b(x')$.

Let's compute the success probability of B:

$$\begin{aligned} \mathsf{Pr}_B^{\text{succ}} &= \mathsf{Pr}[b(x) = b' : x \leftarrow \{0,1\}^n, b' \leftarrow B(1^n, f(x))] = \\ &= \mathsf{Pr}[b(x) = b(x') : x \leftarrow \{0,1\}^n, x' \leftarrow A(1^n, f(x))] \\ &\geq \mathsf{Pr}[x = x' : x \leftarrow \{0,1\}^n, x' \leftarrow A(1^n, f(x))]. \end{aligned}$$

Because the function f is 1-1 we have that $f(x) = f(x')$ if and only if $x = x'$. Thus the above probability equals:

$$\mathsf{Pr}[f(x) = f(x') : x \leftarrow \{0,1\}^n, x' \leftarrow A(1^n, f(x))].$$

This is by assumption non-negligible, contradiction the hypotesis that b is a hard-core predicate.