Homework 25

Joe Baker, Brett Schreiber, Brian Knotten

March 22, 2018

44

a

Given a one-way function f, it is possible to create a new one-way function g which runs in $O(n^2)$ time as follows:
On input x of size n:

```
    Split the input $x$ into $log(n)$ chunks: $x_1, x_2 ... x_{log(n)}$.
    for each $x_i$:
        Compute $f(x_i)$, keeping track of the number of steps $f$ takes. After $n^2$ steps, just output 0.
    Return $f(x_1)||f(x_2)...||f(x_{log(n)})$ where $||$ is the concatenation of the bitstrings.
    Some of these substrings will be 0.
```

First, g runs in $O(n^2)$ time, because f performing $log(n)$ computations. So g is the complexity of f multiplied by $log(n)$. Since we stop f after n^2 steps, the total runtime is $n^2 * log(n) = O(n^2)$.

Second, g is a one way function, since $g = f_U$, and f_U is one-way as proved below.

b

f is one way $\Rightarrow f_U$ is one-way. This can be proved by contrapositive, that f_U is not one-way $\Rightarrow f$ is not one way. Assume f_U is not one-way. Then there exists an algorithm A_U which given y can produce the x such that $f_U(x') = y'$ in polynomial time. Then you can construct an algorithm A which given y can produce the x such that $f(x) = y$ in polynomial time.

$A =$ on input y:

1. Generate $r =$ some number of random bits.
2. Construct the string $y' := y||r$.
3. Run A on y' to get x'.
4. If $f(x') = y$, return x, else, go back to step 1.

A will halt in polynomial time because

45

a

Let (E, D) be a semantically secure encryption scheme and let $f(x)$ be a function that returns 1 if a bit of x is 1 and 0 otherwise. Then, by definition, for all probabilistic poly-time algorithms A: $P(A(E_k(x)) = 1) \leq P(B(1^n) = 1) + \epsilon(n)$. Clearly, if A cannot determine if a bit of $E_k(x) = 1$, then by definition the $P(A(E_k(x) = (i, b)$ s.t. $x_i = b)$ requirement of computational security is satisfied. Further, the probability of a random bit being 1 is $\frac{1}{2}$ so $P(B(1^n) = 1) = \frac{1}{2}$. Then our definition becomes: $P(A(E_k(x)) = 1) \leq \frac{1}{2} + \epsilon(n)$ and semantic security satisfies computational security.

b

Let G be a pseudo-random generator mapping $\{0,1\}^n$ to $\{0,1\}^m$, and (E,D) be an encryption scheme where $E_k(x) = x \oplus G(k)$ and $D_k(y) = y \oplus G(k)$. Since G is a pseudo-random generator, then G can be used to increase the key size to match the message size. When the key size matches the message size, then perfect secrecy can be achieved. Since the encryption function uses an xor scheme, the encrypted message is uninformly distributed over all binary strings of length m. Let A be any probabilistic poly-time algorithm and $x \in_R X_n, k \in_R \{0,1\}^n$.

Consider the algorithm $A(E_{U_n}(0^m))$. Since G is a pseudo-random generator, decrypting $E_k(x)$ will require guessing each bit of x. The probability of guessing 1 of $2^{|x|}$ different bitstrings is uniformly distributed, so an optimal guessing strategy would be guessing all 0's since it is just as likely to be correct as any other guess. Thus $P[A(E_k(x)) = f(x)] \leq P[A(E_{U_n}(0^m)) = f(x) + \epsilon(n)$ and this encryption scheme is semantically secure by definition of perfect secrecy.

c

For the definitions to be equal, it must be proved in both directions.

The general definition to the special case is trivial, since the special case is an assignment to the general definition, namely that X_n is specifically the uniform distribution over a pair of strings x_0^n, x_1^n, and f is the function that maps x_0^n to 0 and x_1^n to 1.

Therefore the bulk of the proof is proving that the specific random variable and the specific function are sufficient for the general definition of semantic security. Since all possible B functions are equally powerful, let $B = A(E_{U_n}(0^m))$.