COM S 6810 Theory of Computing March 31, 2009

Lecture 19: One-Way Functions, Hardness Amplification

Instructor: Rafael Pass *Scribe: Daniel Perelman*

1 One-Way Functions (OWF)

For OWFs, we care about average case, not worst case, hardness.

Recall that a BPP machine makes an error w.p. $\frac{1}{3}$ for a given run, but its worst-case error for any specific x is only $\frac{1}{3}$.

For a OWF f, it should be easy (polynomial time) to compute $f(x)$, but hard to compute its inverse on random x.

Definition 1 (negligible) *A function $\epsilon : N \to R$ is negligible if $\forall c\ \exists n_0$ s.t. $\epsilon(n) < \frac{1}{n^c}\ \forall n > n_0$.*

Definition 2 (strong OWF) *A function $f : \{0,1\}^* \to \{0,1\}^*$ is said to be a (strong) one-way function (OWF) if*

1. *[Easy to comp] $\exists$ PPT that computes f*
2. *[Hard to invert] $\forall$ PPT $A\ \exists$ neg ϵ s.t $\forall n$*

$$\mathsf{Pr}\left[x \leftarrow \{0,1\}^n : A(1^n, f(x)) \in f^{-1}(f(x))\right] \leq \epsilon(n)$$

Example of intuition for a OWF to be hard to invert: given a telephone book, it is easy to find the telephone number for a given name by looking it up, but hard to find a name for a given number as that would require reading through the entire book until hitting upon the number.

Note that $f^{-1}(f(x))$ is used instead of just $\{x\}$ as otherwise truncation would be a OWF function as it would be impossible to recover the truncated bits except by guessing, which would only work with probability exponential in the number of bits truncated, even though truncation would not be very useful as a OWF.

A is given an argument of 1^n in case $f(x)$'s output length is $O(\log n)$ in which case A might not even be able to compute $f(x)$. A could also be defined to be polynomial in the length of x instead of in the length of $f(x)$ as an alternative way of fixing the same problem.

The existence of a OWF implies $NP \neq P$, $NP \not\subseteq BPP$. The implication going the other way is unknown, but believed to be false. In fact, $NP \not\subseteq \not\Longleftarrow \exists$ OWP (one-way permutation – a OWF which is a bijection) unless PH collapses (using Turing reductions).

The multiplication function, $f(x, y) = xy$ where $|x| = |y|$, is an example of a (weak) OWF. It is hard to invert when x, y are prime, but easy to invert when xy has small prime factors.

Definition 3 (weak OWF) *A weak OWF is almost the same as a strong OWF except the hard to invert condition is weakened:*
2′. $\exists$ poly $q(n)$ s.t. $\forall$ PPT A and all sufficiently large n

$$\mathsf{Pr}\left[x \leftarrow \{0,1\}^n : A(1^n, f(x)) \in f^{-1}(f(x))\right] \leq 1 - \frac{1}{q(n)}$$

A weak OWF can be inverted on some high proportion of inputs, but it is hard to invert on the remaining inputs.

2 Hardness Amplification

We have a weak OWF and we want to construct a strong OWF. Basic idea: repeat the weak OWF many times, at least one of them should be hard to invert with relatively high probability. This is non-trivial to prove because "hard to invert" is defined computationally, not information theoretically.

Theorem 1 (Yao) *$\exists$ weak OWF $\Longrightarrow$ $\exists$ strong OWF*

Proof. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a weak OWF and $q(n)$ is the polynomial in (2') of the definition of a weak OWF.

Let $f'(x_1, \ldots, x_m) = (y_1, \ldots, y_m)$ where $y_i = f(x_i)$, $m(n) = 2nq(n)$.

Idea:

1. Assume f' is not OWF with A being a machine which breaks it.
2. Construct A' which inverts f w.p. $1 - \frac{1}{q(n)}$.

Assume $\exists$ PPT A, poly p' s.t. for infinitely many n', A inverts f w.p. $\frac{1}{p'(n')} = \frac{1}{p'(nm)} = \frac{1}{p(n)}$.
$\mathsf{Pr}\left[x_i \leftarrow \{0,1\}^n : A(f'(x_1, \ldots, x_n)) \text{ succeeds}\right] > \frac{1}{p(n)}$

To invert a specific y, generate a list of random strings $y_1, \ldots, y_n$. Place y at random position i of the list, replacing y_i. Simply repeating y may not work because A could simply refuse to invert such strings. Simply putting y in some fixed position may not work because the order could matter for A and, for example, A might not invert most strings put in the first position.

Define $A''(y)$:

1. Pick random $i \in [m]^*$
2. Let $x_j \leftarrow \{0,1\}^n, y_i = f(x_j)$ if $j \neq i$, $y_i = y$
3. $z_1, \ldots, z_n = A(\vec{y})$
4. If $f(z_i) = y$ output z_i
5. Otherwise output $\perp$

$A'(y)$ runs $A''(y)$ $r(n)$ times where $r(n) = 2nm^2p(n)$. Output first answer different from $\perp$. If all runs output $\perp$, output $\perp$.

Let $s = sm^2p(n)$.

x is "good" if $\mathsf{Pr}\left[A''(f(x)) \text{ succeeds}\right] \geq \frac{1}{2m^2p(n)}$, "bad" otherwise.

Claim: # of "good" xs $\geq 2^n\left(1 - \frac{1}{2q(n)}\right)$

Assuming claim, A' fails if x is "bad", which happens w.p. $\leq \frac{1}{2q(n)}$ or x is "good" but A' fails anyway, which happens w.p. $\leq \left(1 - \frac{1}{s}\right)^{sn} \approx \frac{1}{s^n}$

$\mathsf{Pr}\left[A(f(x_1, \ldots, x_n)) \text{ succeeds}\right] = \mathsf{Pr}\left[A(f(x_1, \ldots, x_n)) \text{ succeeds} \wedge \exists x_i \text{ "bad"}\right]$
$+\mathsf{Pr}\left[A(f(x_1, \ldots, x_n)) \text{ succeeds} \wedge \forall x_i \text{ "good"}\right]$

$\mathsf{Pr}\left[A(f(x_1, \ldots, x_n)) \text{ succeeds} \wedge \forall x_i \text{ "good"}\right] \leq \mathsf{Pr}\left[\forall x_i \text{"}good''\right] \leq \left(1 - \frac{1}{2q(n)}\right)^{m=2q(n)n} \approx \left(\frac{1}{e}\right)^n$

$\mathsf{Pr}\left[A(f(x_1, \ldots, x_n)) \text{ succeeds} \wedge \exists x_i \text{ "bad"}\right] \leq \sum_{i=1}^n \mathsf{Pr}\left[A(f(x_1, \ldots, x_n)) \text{ succeeds} \wedge x_i \text{ "bad"}\right] \leq \sum_{i=1}^n \mathsf{Pr}\left[A(f(x_1, \ldots, x_n)) \text{ succeeds} | x_i \text{ "bad"}\right] \leq \sum_{i=1}^n m\mathsf{Pr}\left[A'(x_i) \text{ succeeds} | x_i \text{ "bad"}\right] \leq \sum_{i=1}^n m\frac{1}{2m^2p(n)} = \sum_{i=1}^n \frac{1}{2mp(n)} = \frac{1}{2p(n)} < \frac{1}{q(n)}$

Contradiction. ■