PRINCETON UNIVERSITY COS 522: COMPUTATIONAL COMPLEXITY

Lecture 9: One-way functions and hard-core bit theorem

Lecturer: *Sanjeev Arora* Scribe:*Edith Elkind*

1 Examples and definitions

DEFINITION 1 (INFORMAL DEFINITION) *A family of functions* $\{f_n : \{0,1\}^n \mapsto \{0,1\}^n\}$ *is called* one-way*, if* f_n *are*

- *easy to compute, but*
- *hard to invert for "many" inputs.*

EXAMPLE 1 $f(p,q) = pq$, where pq is n bit long. This function seems hard to invert, when p and q are roughly the same size and $p \equiv q \equiv 3 \pmod 4$. By Prime Number Theorem, about $1/n^2$ of all integers of n bits are products of two such primes.

DEFINITION 2 (FORMAL DEFINITION) *A family of functions* $\{f_n : \{0,1\}^n \mapsto \{0,1\}^n\}$ *is* one-way *with* security $s(n)$ *if there is a polynomial time Turing machine that computes them and furthermore for every algorithm* A *that runs in time* $s(n)$*,*

$$\mathbf{Pr}_{x\in\{0,1\}^n}[A \text{ inverts } f_n(x)] \leq \frac{1}{s(n)}. \tag{1}$$

REMARK 1 Here by inverting a (possibly many-to-one) function we mean finding any preimage of a given element. Also, we can define one-way functions with other classes of adversaries, such as probabilistic Turing machines, or deterministic circuits.

A more general definition of a one-way function would allow the inversion probability of the adversary to be a general function $\delta(n)$, instead of demanding that it should be at most $1/s(n)$. In fact, it may seem risky to use our stringent definition, since a function satisfying it may not exist! For example, the multiplication function in Example 1 is hard to invert only occasionally (on $1/n^2$ of the outputs, as noted) and hence does not satisfy Definition 2 with even $S(n) = n$. Luckily, Yao has proven for us (although we will not show this) that if factoring or some other function is hard to invert on $1/\text{poly}(n)$ fraction of inputs then a one-way function exists that is hard to invert on almost all inputs, and hence satisfies Definition 2 with some $s(n) = \Omega(n^c)$ for every $c > 1$..

2 Goldreich-Levin Theorem

A a one-way function family $\{f_n\}$ is from $\{0,1\}^n$ to $\{0,1\}^n$ is called a *one-way permutation* if each f_n is one-to-one and onto.

EXAMPLE 2 The following is a conjectured one-way permutation. Let $p_1, p_2, \ldots$ be a sequence of primes where p_i has i bits. Let g_i be the generator of the group Z_p^* the set of numbers that're nonzero mod p. Since g_i is a generator, for every $y \in 1, .., p_i - 1$, there is a unique $x \in \{1, .., p-1\}$ such that

$$g_i^x \equiv y \pmod{p_i}.$$

Then $x \to g_i^x \pmod{p_i}$ is a permutation on $1, .., p_i - 1$ and is conjectured to be one-way. The inversion problem is called the *Discrete Log* problem.

Consider two random strings x and r, $|x| = |r| = n$. We use f as a shorthand for f_n. Consider the string $(f(x), r, x \odot r)$, where $x \odot r$ denotes the scalar product of x and r. As f is a permutation, $f(x)$ is information-theoretically equivalent to x, so the last bit of the concatenated string is completely determined by the first $2n$ bits. However, it turns out that this string looks completely random to any reasonable adversary.

THEOREM 1 (GOLDREICH, LEVIN '86)
Suppose that f_n is a one-way permutation and has security $s(n)$. Then for all algorithms A running in time $s^{1/4}(n)$

$$\mathbf{Pr}_{x,r\in\{0,1\}^n}[A(f_n(x), r) = x \odot r] \leq \frac{1}{2} + O(\frac{1}{s(n)}). \tag{2}$$

PROOF: Suppose that A can predict $x \odot r$ with probability $1/2 + \delta$. We show how to invert $f_n(x)$ for $O(\delta)$ fraction of the inputs in $O(n/\delta^2)$ time, from which the theorem follows.

LEMMA 2
Suppose that

$$\mathbf{Pr}_{x,r\in\{0,1\}^n}[A(f_n(x), r) = x \odot r] \geq \frac{1}{2} + \delta. \tag{3}$$

Then for at least δ fraction of x's

$$\mathbf{Pr}_{r\in\{0,1\}^n}[A(f_n(x), r) = x \odot r] \geq \frac{1}{2} + \frac{\delta}{2}. \tag{4}$$

PROOF: We use an averaging argument. Suppose that p is the fraction of x's satisfying (4). We have $p \cdot 1 + (1-p)(1/2 + \delta/2) \geq 1/2 + \delta$. Solving this with respect to p, we obtain

$$p \geq \frac{\delta}{2(1/2 - \delta/2)} \geq \delta.$$

□

We construct an inversion algorithm that given $f_n(x)$, where $x \in_R \{0,1\}^n$, will try to recover x. It "succeeds" with high probability if x is such that (4) holds. Note that the algorithm can always check the correctness of its answer, since it has $f_n(x)$ available to it and it can apply f_n to its answer and see if this gives f_n.

Today, we give the proof for the simpler case (which nevertheless contains all essential ideas) when $\mathbf{Pr}_{r\in\{0,1\}^n}[A(r) = x \odot r] \geq \frac{3}{4} + \delta$. In this case, for each i, $i = 1, \ldots, n$, we pick a random string r and query A to obtain $A(r)$ and $A(r \oplus e_i)$, where e_i is the ith

basis vector, and $\oplus$ denotes addition modulo 2. We know that $A(r)$ is "often" equal to $x \odot r$ and $A(r \oplus e_i)$ is "often" equal to $x \odot (r \oplus e_i)$, so with a high enough probability $A(r) \oplus A(r \oplus e_i) = (x \odot r) \oplus (x \odot (r \oplus e_i) = x \odot e_i = x_i$. More formally, the algorithm guesses that $x_i = A(r) \oplus A(r \oplus e_i)$ and $\mathbf{Pr}_r[\text{the guess for } x_i \text{ is incorrect}] \leq 2(1/4 - \delta) = 1/2 - 2\delta$. (Here we use the fact that if r is random, then $r \oplus e_i$ is random as well, and apply the union bound). We can repeat this experiment sufficient number of times and take the majority vote to amplify the probability of guessing correctly. Furthermore, the probability of guessing the whole word correctly can then be bounded from below by using the union bound once again. $\square$