COM S 6810 Theory of Computing	Apr. 2nd, 2009

Lecture 20: Hard-Core Bits

Instructor: Rafael Pass — *Scribe: Rachel Lin*

As introduced in previous lectures, a one-way function is a function, such that,

- it is easy to compute.
- but hard to invert, meaning, for all PPT adversary A, there exists a negligible function ε, such that, for all $n \in N$,

$$\Pr_x \left[A(1^n, f(x)) \in f^{-1}(f(x))\right] \leq \varepsilon(n)$$

Intuitively, if a function is hard to invert, there should be certain bits in the input x that are hard to compute given $f(x)$. However, this is not ture; consider the following counter example. Let f be a one-way function, for any $0 \leq i \leq n$, define f'_i, such that, $f'_i(x) = x_i \| f(x_0 \dots x_{i-1} x_{i+1} \dots x_{n-1})$. It is obvious that f'_i is also a OWF, but, it leaks the i^{th} bit in the input. This example shows that there does not exist a particular bit of the input that is hard to compute. Hence, we turn to asking:

Does there exist some bit that can be extracted from x, and is hard to compute given $f(x)$?

Definition 1 (Hard Core Bit)**.** *Let f be a one-way function. A predicate $b : \{0,1\}^* \to \{0,1\}$ is a hard core bit for f if*

- *b is efficiently computable,*
- *for all PPT adversary A there exists one negligible function ε, s.t. for all $n \in N$,*

$$Pr_x \left[A(1^n, f(x)) = b(x)\right] \leq \frac{1}{2} + \varepsilon(n)$$

Ideally, we would like to have that

Conjecture 1. every OWF has a hard core bit.

Unfortunately, we don't know if this is ture or not. However, we do know—as shown by Goldreich and Levin—that every one-way function f can be tranformed into another function f', such that, f' is one-way and has a hard core bit.

Theorem 1. *Let f be a OWF. Define function f' and predicate b, such that, $f'(x, r) = f(x) \| r$, and $b(x, r) = \langle x, r \rangle$. Then f' is a one-way function with hard-core bit b.*

Proof. First, it follows directly from the one-wayness of f that f' is a OWF. Thus it only remains to show that b is the hard-core bit of f'. Assume for contradiction that there exists one adversary A that can predict the hard-core bit with high probability. Then we show how to invert the one-way function f. In the following, we first consider two simplified cases, which will provide us the intuition for the final proof.

A Super-simplified Case: Assume that the adversary A predicts the hard-core bit with probability 1. Let $e_i = (00 \dots 1 \dots 0)$ be the n-bit string with the i^{th} bit 1 and all others 0. The following algorithm, on input $y = f(x)$, inverts y with probability 1.

```
B(y): for i = 1 to n
          x_i = A(y, e_i)
      return x;
```

Since A predicts the hard-core bit of f' with probability 1, A, on input (y, e_i), $y = f(x)$, outputs $\langle x, e_i \rangle = x_i$, the i^{th} bit of the input, with probability 1. Thus B inverts f with probability 1.

A Simplified Case: Assume now that A only predicts the hard-core bit with probability $\frac{3}{4} + \frac{1}{p(n)}$, where p is a polynomial. In this case, in order to learn one bit x_i, we cannot directly query A with (y, e_i) as in the algorithm B, since A may always fail whenever the second part of the input is e_i. However, note that $\langle x, e_i \rangle = \langle x, r \rangle \otimes \langle x, r \otimes e_i \rangle$, for any r; we hence instead query A with input (y, r) and $(y, r \otimes e_i)$, and xor the result to learn x_i. We say that an input $x \in \{0,1\}^n$ is good, if it holds that

$$\Pr_r [\ A(f(x), r) = \langle x, r \rangle] \geq \frac{3}{4} + \frac{1}{2p(n)}$$

In other words, conditioned on receiving an input $(f(x), r)$ generated from a good x, A succeeds with very high probability. Then by union bound, for a good x, the probability that A succeeds on both queries $(f(x), r)$ and $(f(x), r \otimes e_i)$, and thus learns x_i, is at least $\frac{1}{2} + \frac{1}{p(n)}$. Then by repeating this process for $m = \text{poly}(p(n))$ times, and taking the majority of the results, we learn x_i with overwhelming probability $(1 - \frac{1}{2^n})$. See algorithm B' below for a more detailed description. Let S_n be the set of good x in $\{0,1\}^n$. It follows from the argument above that:

$$\Pr \left[x \leftarrow S_n \ : \ B'(f(x)) \in f^{-1}(f(x)) \right] \geq 1 - \frac{1}{2^n}$$

```
B'(y): for i = 1 to n
           for j = 1 to m
               pick r at random;
```

```
            c_i = A(y, r) xor A(y, r xor e_i);
        Let x_i be the majority of c_0 to c_m;
    return x;
```

Below lemma 1 shows that there are such good inputs actually constitute a polynomial fraction ($\frac{1}{2p(n)}$) of all inputs. Then we have

$$\Pr_x\left[B(f(x)) \in f^{-1}(f(x))\right] \geq \Pr_x\left[B(f(x)) \in f^{-1}(f(x)) \mid x \text{ good}\right] \Pr_x[x \text{ good}] \geq \frac{1}{3p(n)}$$

This means B' inverts f, which contradicts the one-wayness of f.

Lemma 1. *Then* $|S_n| \geq \frac{1}{2p(n)}2^n$

Proof. Assume for contradiction that $|S_n| < \frac{1}{2p(n)}2^n$. Then $\Pr_x[x \text{ good}] < \frac{1}{2p(n)}$ and thus

$$\begin{aligned}
&\Pr_{x,r}\left[A(1^n, f(x)\|r) = \langle x, r\rangle\right] \\
=&\Pr_{x,r}\left[A(1^n, f(x)\|r) = \langle x, r\rangle \middle| x\text{good}\right] \Pr_x[x \text{ good}] \\
&+ \Pr_{x,r}\left[A(1^n, f(x)\|r) = \langle x, r\rangle \middle| x \text{ not good}\right] \Pr_x[x \text{ not good}] \\
<&\frac{1}{2p(n)} + \Pr_{x,r}\left[A(1^n, f(x)\|r) = \langle x, r\rangle \;\middle|\; x \text{ not good}\right] \Pr_x[x \text{ not good}] \\
<&\frac{1}{2p(n)} + \frac{3}{4} + \frac{1}{2p(n)}
\end{aligned}$$

Final Case: Now we are ready to move on to the final case where A only predicts the hard-core bit of f' with probability slightly more than $\frac{1}{2}$, that is $\frac{1}{2} + \frac{1}{p(n)}$. First, note that, in this case, the algorithm B' fails, since the fraction of good x among all inputs can be very small. As a remedy, we loosen the requirement of good x; we say that a x is okay, if it holds that

$$\Pr_r\left[A(f(x), r) = \langle x, r\rangle \geq \frac{1}{2} + \frac{1}{2p(n)}\right]$$

It follows from essentially the same proof that

Claim 1. *Let T_n denote the set of* okay *inputs in* $\{0,1\}^n$. *Then* $|T_n| \geq \frac{1}{2p(n)}2^n$

However, unlike inputs that are good, for an okay x, the probability that A succeeds on both queries, $(f(x), r)$ and $(f(x), r \otimes e_i)$, can be much smaller than $\frac{1}{2}$. Then running B' on input generated from an okay x does not yield x_i with high probability. This problem can be solved, assuming that the algorithm has access to an oracle $\mathcal{O}$, such that, $\mathcal{O}$ on input $f(x)$ returns $m = \text{poly}(p(n))$ i.i.d. samples, $(r_1, \langle x, r_1\rangle)$, $\ldots$, $(r_m, \langle x, r_m\rangle)$. Given those samples, to learn x_i the algorithm only need to query A with inputs $(f(x), r_k \otimes e_i)$, which will succeed with probability $\frac{1}{2} + \frac{1}{p(n)}$, if x is okay.

```
B''(y): Query O on y; obtain (r_j, z_j), z_j = <x, r_j> for j from 1 to m
        for i = 1 to n
            for j = 1 to m
                c_ij = z_j  xor A(y, r_j xor e_i);
            Let x_i be the majority of c_i0 to c_im;
        return x;
```

Since each c_{ij} is correct (equals to x_i) with probability at least $\frac{1}{2} + \frac{1}{p(n)}$. The majority of c_{i0} to c_{im} would be correct with overwhelming probability, according to Chernoff bound. However, notice that for the majority to be correct with overwhelming probability, we do not need c_{i0} to c_{im} to be all independent; in fact, it is sufficient to have them pairwise independent (by Chebyshev's inequality). In other words, it is sufficient to have an oracle returning only pairwise independent samples. Based on this observation, below we show how to simulate the oracle.

On input $f(x)$, sample randomly s_1 to s_l, where $l = \log m$, and guess, y_1 to y_l, such that $y_i = \langle x, s_i \rangle$. Since there are only $\log m$ samples, the probability that all y_i are guessed correctly is $\frac{1}{m}$. Next, to generate pariwise independent samples, take all possible subset S of $\{0, \ldots, m\}$, and compute $r_S = \otimes_{i \in S} s_i$ and $z_S = \otimes_{i \in S} y_i$. It is clear that all r_s are pairwise independent, and if all y_i are guessed correctly, $z_s = \langle x, r_S \rangle$ for all S. Therefore, we can simulate the oracle $\mathcal{O}$ correctly with probability $\frac{1}{m}$. Finally, putting all the pieces together, we obtain the final algorithm:

```
B'''(y): Simulate O; obtain (r_j, z_j), z_j = <x, r_j> for j from 1 to m
         for i = 1 to n
            for j = 1 to m
                c_ij = z_j  xor A(y, r_j xor e_i);
            Let x_i be the majority of c_i0 to c_im;
         return x;
```