CSE 594 : Modern Cryptography 02/13/2017

Lecture 6: The Goldreich Levin Theorum

Instructor: Omkant Pandey *Scribe: Hemant Pandey, Sayan Bandyopadhyay*

1 Last Class

In the last class we studied about hardcore predicates for a one way function, Goldreich Levin theorem, Markov's inequality. Today, we are going to discuss Goldreich Levin Theorem and its proof in details along with an overview of Chebychev's inequality.

2 Goldreich Levin Theorem

The context of Goldreich and Levin is to find a hard-core predicate for any one-way function. Let's recall first what a hard core predicate was :

A predicate $h : \{0,1\}^* \to \{0,1\}$ is a hard core predicate for f if h is deterministic and efficiently computable given x and there exists a negligible function ν such that for every non uniform PPT adversary A and for all (sufficiently large) $n \in \mathbb{N}$:

$$\Pr[x \leftarrow \{0,1\}^n : A(1^n, f(x)) = h(x)] \leq \frac{1}{2} + \nu(n)$$

Let us recall the outline of Goldreich Levin Theorem which was discussed in the previous class: Let f be a OWF (OWP). We defined the function $g(x,r) = (f(x), r)$ where, $|x| = |r|$. It is not hard to see that g is also a OWF (OWP). The Goldreich-Levin Theorem proves that $h(x,r) =< x, r >$ is a hard core predicate for g.

This means that if there was an efficient algorithm to predict $< x, r >$ given $g(x,r)$ (which is equal to $(f(x), r)$) there is also an algorithm to compute pre-image of $f(x)$ given $f(x)$. Probabilities over here are taken over random choice of x and r. In previous lecture we did two warm up proofs. We showed that if an algorithm can predict $< x, r >$ with a little over 3/4 probability, we can invert f with noticeable probability. Today, we will prove that this works even if the probability is just above 1/2.

In order to do this, we first need to understand a few concepts.

2.1 Pairwise Independence

We say that $X_1 ... X_M$ are pairwise independent if for every i, j $\in$ [M] with i $\neq$ j and every a, b $\in$ R we have,

$$\Pr[X_i = a \wedge X_j = b] = \Pr[X_i = a] \cdot \Pr[X_j = b]$$

For pairwise independent variables, the following important equation holds:

$$E[Xi \cdot Xj] = E[Xi] \cdot E[Xj].$$

We can use this property with the Chebyshev inequality in the last class to get meaningful bounds for sums of pairwise independent 0/1-random variables.

2.2 Chebyshev's inequality for Sum of Pairwise Independent Boolean Variables

Recall from last class that if Y is a random variable with variance (denoted from hereon as) $V[Y]$, then by Chebyshev's inequality: $\Pr[|Y - E[Y]| > k] \leq \frac{V[Y]}{k^2}$.

Now, suppose that we have n 0/1-random-variables denoted by $X_1, \ldots, X_m$ such that: $\Pr[X_i = 1] = p$. We want to get a meaningful bound for a random variable that is sum of them all. That is, if we denote the sum by:

$$X = X_1 + X_2 + \ldots + X_m$$

Then we want a meaning bound for how far X can deviate from its expected value.

Note that each X_i has expected value: $E[X_i] = p.1 + (1-p).0 = p$. Therefore, by linearity of expectation:

$$E[X] = E[\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} p = mp$$

We want to know what is the probability that X will be "far" from its expected value mp? In particular, what the is the probability that X is δm far from its expectation? In other words, we want to know a bound on $\Pr[|X - mp| > m\delta]$. Let us write $\mu = E[X] = mp$. We claim that:

$$\Pr[|X - \mu| > m\delta] \leq \frac{1}{4m\delta^2} \tag{1}$$

Proof: We will apply Chebyshev. To do so, we first calculate the variance $V[X]$. We apply the formula for the variance:

$$\begin{aligned}
V[X] &= E[(X - E[X])^2] \\
&= E[(X - \mu)^2] \\
&= E[(X^2 - 2\mu X + \mu^2)] \\
&= E[X^2] - E[2\mu X] + E[\mu^2] \\
&= E[X^2] - 2\mu E[X] + \mu^2 \\
&= E[X^2] - 2\mu^2 + \mu^2 \\
&= E[X^2] - \mu^2 \\
&= E[X^2] - m^2p^2
\end{aligned}$$

Now, let's calculate $E[X^2]$. First, notice that for each X_i: $E[X_i^2] = p.1^2 + (1-p).0^2 = p$ and for each $i \neq j$: $E[X_i \cdot X_j] = E[X_i] \cdot E[X_j$ because X_i, X_j are pairwise independent. Therefore,

$$
\begin{aligned}
E[X^2] &= E[(X_1 + X_2 + \ldots.. + X_m)^2] \\
&= E[\sum_{i,j} X_i \cdot X_j] \\
&= E[\sum_{i \neq j} X_i \cdot X_j] + \sum_i E[X_i^2] \\
&= \sum_{i \neq j} E[X_i] \cdot E[X_j] + \sum_i p \\
&= \sum_{i \neq j} p \cdot p + mp \\
&= p^2 \cdot (m^2 - m) + mp \\
&= m^2 p^2 + mp(1-p).
\end{aligned}
$$

Substituting the value of $E[X^2]$ back in the calculation for $V[X]$ we get:

$$V[X] = mp(1-p).$$

Now substituting the value of $V(x)$ back in Chebyshev's inequality, we can get equation (1). That is:

$$
\begin{aligned}
\Pr[|X - \mu| > m\delta] &\leq \frac{V[X]}{(m\delta)^2} \\
&= \frac{mp(1-p)}{m^2\delta^2} \\
&= \frac{p(1-p)}{m\delta^2} \\
&\leq \frac{1}{4m\delta^2}
\end{aligned}
$$

where the last line follows from the fact that $p(1-p)$ attains its maximum value when $p = 1 - p = \frac{1}{2}$. This proves equation (1) which we will use soon.

2.3 From few independent bits to many pairwise independent bits

Suppose that b_1, b_2 are independent random bits. Then the tuple (b_1, b_2, b_3) where $b_3 = b_1 \oplus b_2$ is a tuple of 3 *pairwise independent* bits. This fact is easy to verify by simply using the definition of pairwise independent.

We can extend this concept to many bits. In particular, we are given l random and independent bits: $(b_1, b_2, b_3 \ldots\ldots. b_l)$ we can create m bits from them: $(b'_1, b'_2 \ldots\ldots. b'_m)$ such that these m bits are pairwise independent and $m = 2^l - 1$. Indeed, observe that there are $2^l - 1$ *non-empty* subsets of the set $[l] = \{1, 2, \ldots, l\}$. Therefore, we can define a bit for each of these subsets as follows: $b'_S = \oplus_{i \in S} b_i$ for $S \subset [l]$ and $S \neq \phi$. So if we number these sets from 1 to m, our bits will be $(b'_1, \ldots, b'_m)$. It is easy to check that these m bits are pairwise independent as before.

Why is this important? The above fact is important for the following reason. Suppose that someone comes and gives us *correct* hardcore bits for l challenges, then we can generate hard core bits for m challenges simply by xoring as above. This works only for the inner product function $< x, r >$ and may not work for other types of hard core predicates.

Now, how will we get these l values? Well, if we set $l = \log n$, then we can just guess them randomly and we will be correct with probability at least $1/n$. And then by applying the above method, we can get $m = 2^l - 1$ correct values without guessing. So we get m values by guessing only l which means we get m correct values with probability $1/n$. Now lets use this.

2.4 Proof of GL Theorem

Given A such that :

$$Pr_{x,r}[A(f(x), r) =< x, r >] \geq \frac{1}{2} + \epsilon \tag{2}$$

We will design an algorithm B for inverting f with probability more than $\epsilon/4$.

To do this, let us first define a good set of x values. These are the x values for which A guesses the hardcore bit with better than $1/2$ probability. (Note that not all x have this property — we only know that A *on average* guesses hardcore bits for a random x with better than $1/2$; so we want to define a set where we are guaranteed that A always has good chance as we change r but keep x fixed). Let Gd be the set of good values defined as follows:

$$Gd = \left\{ x : Pr_r[A(f(x), r) =< x, r >] \geq \frac{1}{2} + \frac{\epsilon}{2} \right\}$$

We claim that there are many good x values; more precisely:

$$\Pr_x[x \in \mathit{Good}] \geq \frac{\epsilon}{2} \tag{3}$$

Proof: Suppose that this is not true, i.e.: $\Pr_x[x \in \mathit{Good}] < \frac{\epsilon}{2}$.

Then,

$$\begin{aligned}
\Pr_{x,r}[A(f(x), r) =< x, r >] &= \Pr_{x,r}[A(f(x), r) =< x, r > | x \in Gd] \cdot \Pr_{x,r}[x \in Gd] \\
&\quad + \Pr_{x,r}[A(f(x), r) =< x, r > | x \notin Gd] \cdot \Pr_{x,r}[x \notin Gd] \\
&< 1 \cdot \frac{\epsilon}{2} + (\frac{1}{2} + \frac{\epsilon}{2}) \cdot 1 \\
&= \frac{1}{2} + \epsilon.
\end{aligned}$$

This is a contradiction to the assumption that A guesses $< x, r >$ with $1/2 + \epsilon$ probability or more. Hence the claim.

Now we define adversary B which guesses $b_1, b_2, ...b_l$ for random values $r_1, r_2, ...r_l$ and then generates values $b'1, \ldots, b'_m$ and $r'_1, \ldots, r'_m$ as we discussed above. And then uses them to guess bits of x one by one.

Here the main idea to guess each bit of x. Suppose that the values B generates are correct hard core bits: i.e., $(b'_1, \ldots, b'_m)$ and $(r'_1, \ldots, r'_m)$ are such that $< x, r'_j >= b'_j$. Then, the B can use A to guess the hardcore bit for $r''_j = e_i \oplus r'_j$. It can then recover a guess for x_i as we did in the warm up proof for $3/4 + \epsilon$ case. That is, define:

$$x^*_{i,j} = b'_j \oplus b''_{i,j} \text{ where } b''_j = A(f(x), e_i \oplus r'_j).$$

Then, the guess for x_i is obtained as:

$$x^*_i = \text{majority bit in } \{x^*_{i,j}\}_{j=1}^m$$

We claim that if $m = \frac{2n}{\epsilon^2}$ then for every $x \in Gd$:

$$\Pr[x_i^* \neq x_i] < \frac{1}{2n} \tag{4}$$

Proof: Keep an indicator variable y_j such that $y_j = 1$ if $x_{i,j} \neq x_i$. Let:

$$y = y_1 + y_2 + ... + y_m$$

Then, x_i^* is not correct if $y > m/2$. We apply Chebyshev for $x \in Gd$. Notice that for $x \in Gd$ each y_i is 1 with probability $p = \Pr[y_i = 1] = 1 - (\frac{1}{2} + \frac{\epsilon}{2}) = \frac{1-\epsilon}{2}$ and $E[y] = mp$ where $m = 2n/\epsilon^2$. Let $\delta = 1/2 - p = \epsilon/2$. Then using Chebyshev:

$$\begin{aligned}
\Pr[y > m/2] &= \Pr[y - mp > m/2 - mp] \\
&\leq \Pr[|y - E[y]| > (1/2 - p)m] \\
&< \frac{1}{4m\delta^2} \\
&= \frac{1}{4 \cdot \frac{2n}{\epsilon^2} \cdot \frac{\epsilon^2}{4}} \\
&= \frac{1}{2n}
\end{aligned}$$

As claimed. Therefore, for any given x, by the above strategy B will get x_i wrong for any given $x \in Gd$ with at most $1/2n$ probability. By union bound, if B guesses each x_i one by one for each i to construct full x, the probability that x will not be correct is at most $n \times 12n = 1/2$. This gives us the following algorithm B for inverting $f(x)$ for a random x:

Algorithm B to invert f: On input a challenge $z = f(x)$ for a random x do the following:

1. Pick random values $(r_1, \ldots, r_l)$ for $l = \log m + 1$ where $m = \frac{2n}{\epsilon^2}$.
2. Cycle through all possible values of $(b_1, \ldots, b_l)$ starting from $(0, 0, \ldots, 0)$ to $(1, 1, \ldots, 1)$ doing the following:

   for $i = 1$ to n:

   (a) Construct strings $(r'_1, \ldots, r'_m)$ and $(b'_1, \ldots, b'_m)$ using the set construction as defined above.

   (b) for $j = 1$ to m: feed $e_j \oplus r'_j$ to A and get his answer, denoted: $b''_j = A(f(x), e_i \oplus r'_j)$.

   (c) Compute $x_i^* =$ majority bit in $\{x_{i,j}^*\}_{j=1}^m$ where $x_{i,j}^* = b'_j \oplus b''_{i,j}$.

   return x^* if $f(x^*) = z$ where $x^* = x_1^*, \ldots, x_n^*$.
3. Return fail. (i.e., no candidate x^* found so far).

It is easy to check that B runs in polynomial time. We have already argued that if $x \in Gd$ then the probability that B is wrong about x^* is at most $1/2n$ provided that it starts with $(b_1, \ldots, b_l)$ that are correct hardcore bits corresponding to $(r_1, \ldots, r_l)$. Since B cycles through all possible values of $(b_1, \ldots, b_l)$, one of them would be correct. Therefore, when the loop in point 2 exits, the probability that B does not invert z for any $x \in Gd$ is at most $1/2$. Since x is chosen uniformly, it is in Gd with probability at least $\epsilon/2$ as we argued before. Therefore, B inverts f with probability at least $\epsilon/2 \cdot 1/2 = \epsilon/4$. This is a contradiction and proves the GL theorem.