Build a provable secure PRG

Introduction about PRG

PRG means Pseudo random generator. Before we start let's discuss what is Pseudo randomness. It refers a distribution on strings, and when we say that a distribution D over strings of length l is pseudo-random this means that D is indistinguishable from uniform distribution over strings of length l. To be precise, It is infeasible for any polynomial-time algorithm to tell whether it is given a string sampled according to D or an l-bit string chosen uniformly at random.

But what type of distributions should we consider here?

We define D by choosing a short random seed <- $\{0,1\}^n$ uniformly at random and then outputting G(s) belongs to $\{0,1\}^l$ and probability is

$$\frac{|\, s \,\epsilon\{0,1\}^n \,\{\, G(s) = y\}|}{2^n}$$

We will only be interested in the case of l > n in which case the distribution will be far from uniform.

Definition

Pseudorandom generator: It is a deterministic algorithm that receives a seed and produces some longer length string than seed which is pseudorandom.

Let us take the length of input seed as n then the output of PRG is at least n+1 means expansion factor $l(n) = n + 1$.

Design

How to design Pseudorandom generator?

To Design PRG we need some assumptions and required minimal assumption is One way function. One way function is easy to computer but hard to invert it.So we are interested in tasks that are hard to invert in polynomial time algorithms, except with negligible property.

What are suitable candidates for One-way functions?

We can consider the **Factorization problem, Subset-sum problems, Discrete Logarithmic problem** here.

Factorization problem: Finding prime factorization of large prime number is not possible in polynomial time.

Subset sum problem: Given a subset and a subset sum value it is computationally not possible to find subset in polynomial time.

Now, here we are using **Discrete Logarithm problem** (DLP) as Oneway function.

Discrete Logarithm problem: Given values a, b, and n where n is a prime number it is extremely easy to compute $g(b) = a^b \ mod \ n$ but if you have values a, n, and x then it is hard to compute b.

Hardcore predicates:

A hardcore predicate hc of a function f is a function outputting a single bit with the following property:

If $f(x)$ is one-way function, then upon input $f(x)$ it is infeasible to correctly guess $hc(x)$ with any non-negligible advantage above ½.(Note that it is always possible to compute $hc(x)$ correctly with probability ½ by just randomly guessing it)

Suitable candidates for hardcore predicates: Let f be a one-way function and define $g(x,r) = (f(x),r)$. Then , the function $hc(x)$ can be

- MSB(r) or LSB(r)
- Performing AND operation between each bit of x,r and XORing every AND operation results

If we choose first option as a hardcore predicate then the Pseudorandom generator becomes with expansion factor $l(n) = n + 1$