PRG is a deterministic polynomial time algorithm which takes input of n-bits and outputs l(n) bits where:

- l(n) > n
- Output of PRG is computationally indistinguishable from uniform distribution.

One-time pad requires a key of size atleast the size of the message. This fulfils the requirement of perfect secrecy but it can be used only once. By using PRG we can use a much smaller key and yet encrypt a bigger message. We can design a secure encryption scheme using PRGs by pseudo-randomizing the one-time pad.

We will be generating the n-bit key uniformly at random from $\{0,1\}^n$. If the message is of n bits then the one-time pad itself can be used. But if the message is of l(n) bits then we will have to pseudo-randomize the one-time pad. We will take PRG that takes n-bit key as the seed and expands it to l(n) bits output which is random looking, i.e., no efficient adversary can distinguish whether it is uniformly chosen randomly as l(n)-bits or the seed is chosen randomly and expanded with the PRG.

Assume that there exists a PRG with expansion factor l(n) = n+1. Then for any polynomial p(.), there exists a pseudorandom generation with expansion factor l(n) = p(n). We can use a one-way function to design a single-bit expansion PRG. These are the functions that are easy to compute but hard to invert. Discrete logarithm problem is a one-way function.

$\mathbf{f_{p,g}(x) = g^x \bmod p}$, where p is a prime number

A function h_c is a hardcore predicate of one-way function f if:

- h_c can be computed in polynomial-time
- For every probabilistic polynomial time algorithm A, there exists a negligible function negl such that

$\mathbf{P[A(f(x) = h_c(x)] <= 0.5 + negl(n)}$, x is from $\{0,1\}^n$

MSB(x) is a hardcore predicate of discrete logarithm problem.

Let f be a one-way permutation and let h_c be a hard-core predicate of f, then,
$\mathbf{G(x) = (f(x),\ h_c(x))}$ constitutes a PRG with expansion factor l(n) = n+1. In case of discrete logarithm, we have $\mathbf{G(x) = (g^x \bmod p, msb(x))}$

For implementing our PRG, we have considered the following:

- A PRG H, which takes input of n-bits and output n+1 bits.
- The first half and the second half of the seed is given as inputs to the function H such that it returns f**(s_{left}) || s_{right} || hardcore_bit**, where “**||**” indicates concatenation and f is the DLP. Length of f(s_{left}) and s_{right} is n/2 and n/2 respectively and the hardcore_bit is of length 1. Together they form n+1 bits.
- The $(n+1)^{th}$ bit or the hard core bit is extracted and appended to the final output string.
- According to Goldreich-Levin theorem, if a function f(x) is one-way function, ang g(x,y) = f(x) || y, then <x,y> is a hardcore bit for the function g.
- The hardcore bit <x,y> can obtained by performing XOR operation with x_i & y_i for i=0 to $|x_i|$ - 1.
- We iterate for l(n) times to obtain one bit each time the algorithm runs such that our final output is of l(n) bits.