Build a provably secure PRG

A deterministic polynomial time algorithm G, inputs n bits and outputs l(n) bits where:

a. l(n) > n, and
b. Output of G is computationally indistinguishable from uniform distribution

In other words, let l(.) be a polynomial and let G be a deterministic polynomial-time algorithm such that for any input s belonging to $\{0,1\}^n$, algorithm G outputs a string of length l(n). We say that G is a pseudorandom generator if the following two conditions hold:

1. (Expansion:) For every n it holds that l(n) > n.
2. (Pseudorandomness:) For all probabilistic polynomial-time distinguishers D, there exists a negligible function negl such that:

   $|Pr[D(r) = 1] = Pr[D(G(s)) = 1]| <= negl(n)$,

   Where r is chosen uniformly at random from $\{0,1\}^{l(n)}$, the seed s is chosen uniformly at random from $\{0,1)^n$, and the probabilities are taken over the random coins used by D and the choice of r and s.

The function l(.) is called the expansion factor of G.

There are three ways of designing PRGs from computational hardness.

1. Single bit expansion PRG to arbitrary expansion PRG
2. From one-way functions to single-bit expansion PRG
3. Candidate PRG from Discrete Logarithm

Following Step 3, Let f be a one-way permutation and let hc be a hardcore predicate of f. Then, G(s) = (f(s), hc(s)) constitutes a pseudorandom generator with expansion factor l(n) = n + 1.

In case of discrete logarithms : $G(s) = (g^s \bmod p, msb(s))$

Here, msb(x) is a hardcore predicate of Discrete Logarithm Problem and it is the hardest bit of information about x to obtain from f(x).

A function hc : {0,1}* -> {0,1} is a hard-core predicate of a function f if

1. hc can be computed in polynomial time, and
2. For every probabilistic polynomial-time algorithm A there exists a negligible function negl such that

   $Pr\ (x \leftarrow \{0,1\}^n)\ [A(f(x)) = hc(x)] <= ½ + negl(n)$,

   Where the probability is taken over the uniform choice of x in $\{0,1\}^n$ and the random coin tosses of A.