Pseudo Random Generator:

Def^n: A deterministic polynomial time algorithm G, input n bits and output $l(n)$ bits where:

a> $l(n) > n$

b> Output of G is computationally indistinguishable from uniform distribution.

Indistinguishable means that there is no algorithm executable in polynomial time on a probabilistic Turing machine that can decide if the given sequence is random or calculated.

Algorithm:- Let G(PRG) be a function such that

$$G : \{0,1\}^k \rightarrow \{0,1\}^{l(k)}$$

with l as a monotonically increasing function. That means output is always longer than the input (seed).

Also for every algorithm in D belonging to BPP class, for every polynomial p and for every integer k sufficiently big:

$$\left| Pr[x \leftarrow \{0,1\}^k ; r \leftarrow G(x) : D(r) = 1] - Pr[r \leftarrow \{0,1\}^{l(k)} ; D(r) = 1] \right| < \frac{1}{p(k)}$$

i.e. the probability that an algorithm in the class of probabilistic polynomial time problems (BPP) could distinguish a sequence between a real random source and a PRG tends to 0 faster than any polynomial as length of seed increases.

Buliding G

Starting from a simple PRNG H

$H: \{0,1\}^k \rightarrow \{0,1\}^{k+1}$

it is possible to build any PRNG in the form of G as follows:

function $G(x_0)$:

$x_1 \cdot \lambda_1 = H(x_0)$

$x_2 \cdot \lambda_2 = H(x_1)$

$\vdots$

$x_{l(k)} \cdot \lambda_{l(k)} = H(x_{l(k)-1})$

return $\lambda_1 \cdot \lambda_2 \cdots \lambda_{l(k)}$

[x_i is a k-bit string and λ_i is a single bit].

$x_i \cdot \lambda_i$: bitstring resulting after the concatenation of x_i and λ_i.

The H function generates one bit longer sequence from initial seed. By calling the H function $l(k)$ times and taking just the last bit from each iteration, we have generated a sequence of $l(k)$ bits.

~~Proving Secure~~

Provable Secure:-

H function is generally choosen to be a one-way permutation, i.e. a function which is hard to invert given y.

A widely used one-way permutation is modular exponentiation. Given a prime number p and an integer x such that $0 < x < p-1$

$$f(x) = g^{x} \bmod p$$

where g is a generator for the cyclic group $Z_p^{\times}$

The number of generators of this cyclic group is

$$\phi(\phi(p)) = \phi(p-1)$$

p: odd prime ϕ: Euler's totient function

Now finding such x requires the computation of the discrete logarithm, which is a famous unsolved computational problem.

That is to say, no efficient method to calculate the discrete logarithm of a big integer is known.

Hence, we proved the security of PRG by taking DLP(Discrete log assumption).