**Cryptographically Secure Pseudorandom Number Generators (CSPRNGs)**

**Commitment using a Pseudorandom Number Generator**

**The Definition of G(R)**

If R is a true random n-bit seed, we define G(R) to be the 3n-bit number (R, R’, R”), where

R’ ≡ gR (mod p), R” ≡ gR’ (mod p), and p is an n-bit safe prime with primitive root g.

**Commitment Protocol**

\*Bob selects a true random 3n-bit number S and sends S to Alice.

\*Alice selects a true random n-bit number R and computes the 3n-bit number G(R).

\*If b=1 Alice sends G(R) to Bob, otherwise she sends the bitwise XOR, G(R) XOR S to Bob.

**Revelation Protocol**

\*Alice sends R to Bob, who can then check whether he initially received G(R) or S XOR G(R).

**Security**

Alice’s commitment derives from she is not able to find a different random number T such that

G(T) = S XOR G(R) or such that S XOR G(T) = R.

In more detail, the scheme is statistically binding, meaning that even if Alice has infinite computing power, she cannot cheat with probability greater than 2-n. For Alice to cheat, she would need to find a R’, such that G(R') = G(R) XOR S. If she could find such a value, she could decommit by sending the truth and R, or send the opposite answer and R'. However, G(R) and G(R') are only able to produce 2n possible values each (that's 22n) while S is picked out of 23n values. She does not pick S, so there is a 22n/23n = 2-n probability that a R' satisfying the equation required to cheat will not exist.

Bob is not able to figure out whether b is 1 or 0 before the revelation, because that would mean that he could tell the difference between G(R) and G(R) XOR S. Now S is a true random number, and so

S XOR G(R) is a true random number. So this would mean that Bob could tell the difference between G(R) and a true random number. But we are assuming that G(R) is good enough, that there is no test that Bob could apply to tell the difference.

**Remarks**

The security of G(R) is derived from the difficulty of solving the discrete logarithm problem when p is a safe prime. One can increase the security by making p a doubly safe prime. In other words, p = 2q + 1, where q is a safe prime. That is, q = 2r + 1, where r is also a prime. One could similarly define triply safe primes, etc.

**The Discrete Exponential Random Number Generator**

Let p be a large prime and g a primitive root, and let the sequence {xn} where x0, the seed, is a true random number, and rule of succession is x’’ = gx (mod p). That is, if xn = x, then xn+1 = x’.

It was proved originally that rn = (xn)%2 is a secure sequence of bits, in the sense that if the discrete log problem cannot be solved, then rn  cannot be predicted with probability better than ½ + o(1/log n).

This has now been improved considerably, so that if p is a safe prime, then the least n – ω(log n) significant of the n = log p bits will be essentially random. The security in this case is ensured by the discrete log problem for small exponents, or DLPSE, which says that solving for B, given b, in

b ≡ gB (mod p) is intractable even when B is a ω(log n) bit number. The DLPSE is believed to be so if p is a safe prime.