Soumodipta Bose| 2021201086

Build a Provably Secure PRG(Theory)

Discrete logarithm problem:

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

Here p is our prime number of the group and g is the generator. x is the initial seed.

We are assuming that discrete logarithm problem is a one-way function, that is it is easy to compute but hard to Invert. So we are using this in our discrete log function.

DEFINITION 6.1 *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is* one-way *if the following two conditions hold:*

1. (Easy to compute:) *There exists a polynomial-time algorithm* M_f *computing* f*; that is,* $M_f(x) = f(x)$ *for all* x*.*
2. (Hard to invert:) *For every probabilistic polynomial-time algorithm* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \leq \mathsf{negl}(n).$$

[1]

Once we find the generated number, we will need to select the Hardcore bits. Ordinarily we would select the MSB of the generated number from the discrete log problem. But to achieve a little more randomness I have selected the Hardcore Bit Selection using one of the examples of Blum Micali's discrete log problem.

Here what we do is that we we select the msb(x) based on whether our number is greater or less than half of the prime number that was selected and return 0 or 1 based on that and we keep on appending this bit to the output to generate our pseudo-random number.

Let $f : \{0,1\}^n \to \{0,1\}^n$ be a permutation. A (t, ϵ)-hardcore bit is a function $B : \{0,1\}^n \to \{0,1\}$ (predicate) such that for any t-time algorithm M we have

$$|\Pr[M(f(X)) = B(X) | X \leftarrow \{0,1\}^n] - 1/2| < \epsilon$$

("$B(x)$ is hard to compute from $f(x)$.")

Examples:

- Hardcore bit of discrete log [Blum-Micali '81]: Let p be an n-bit prime, $g \in \mathbb{Z}_p^*$. Define $B : \mathbb{Z}_p^* \to \{0,1\}$ as

$$B(x) = msb(x) = \begin{cases} 0 & \text{if } x < p/2 \\ 1 & \text{if } x > p/2 \end{cases}$$

[2]

In the above reference epsilon is our negligible function.

> ***THEOREM 6.7*** *Let f be a one-way permutation and let* hc *be a hard-core predicate of f. Then, $G(s) = (f(s), \mathsf{hc}(s))$ constitutes a pseudorandom generator with expansion factor $\ell(n) = n + 1$.*

In our implementation we used the function g(x) to do these operations, so g(x) has an expansion factor of L(n)=n+1

> ***THEOREM 6.8*** *Assume that there exist pseudorandom generators with expansion factor $\ell(n) = n + 1$. Then, for every polynomial $p(\cdot)$, there exists a pseudorandom generator with expansion factor $\ell(n) = p(n)$.*

We keep on doing this for number of times specified by our polynomial function. If the polynomial is p(x)=2*x then our pseudo random generator runs for 2n times and generates hardcore bits for 2n times and so the output becomes 2n bits.

References

[1] J. K. a. Y. Lindell, Introduction to Modern Cryptography.

[2] B. Micali, "Hardcord bits," [Online]. Available: https://crypto.stanford.edu/pbc/notes/crypto/hardcore.html.