2021201041
Nvss Maneesh Gupta

Discrete Logarithm Function (DL)

Taking p as a prime, g is a generator such that $g \in Z_p^*$ and x as the initial seed.

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

Assuming discrete logarithm is a one-way function.

A function $f: \{0,1\}^* \rightarrow \{0,1\}^*$ 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 A, there exists a negligible function $negl$ such that
$$\Pr\left[Invert_{A,f}(n) = 1\right] \leq negl(n)$$

Hardcore Predicate

A function $hc: \{0,1\}^* \rightarrow \{0,1\}$ 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
$$Pr_{x \leftarrow \{0,1\}^n}\left[A\left(f(x)\right) = hc(x)\right] \leq \frac{1}{2} + negl(n)$$
where the probability is taken over the uniform choice of x in $\{0,1\}^n$ and the random coin tosses of A.

Pseudo Random Generator (PRG)

PRG is obtained by taking DL as a one-way function and the hard-core predicate of the one-way function. Let f be a one-way permutation and let hc be a hard-core predicate of f. Then, $G(s) = (f(s), hc(s))$ constitutes a pseudorandom generator with expansion factor $l(n) = n + 1$.

In case of DL, MSB(x) is a hardcore predicate. Hence, the function G is as follows
$$G(s) = (g^s mod\; p, msb(s))$$
Hardcore bit of DL

Let p be an n-bit prime, $g \in Z_p^*$. Define $B: Z_p^* \rightarrow \{0,1\}$ as $B(x) = msb(x) = \begin{cases} 0 \; if \; x < p/2 \\ 1 \; if \; x > p/2 \end{cases}$

B(x) is hard to compute from f(x), where f(x) is the DL function.

The above function G, can generate an expansion from n bit to n+1 bit. Assuming that there exists a pseudorandom generator with expansion factor $l(n) = n + 1$ them for any polynomial p(.), there exists a pseudorandom generator with expansion factor $l(n) = p(n)$.

1. Take the last bit from l+1 length string for output.
2. Consider the remaining l length output as input of G for the next iteration.
3. Apply above 1,2 steps n times to get output of string n

2021201041
Nvss Maneesh Gupta

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.

[3] Lecture Slides