1 Discrete Logarithm Problem

1.1 The multiplicative group modulo p

The multiplicative group modulo p is the set of $p-1$ elements $\{1, 2, \ldots, p\text{-}1\}$ under the group operation multiplication modulo p, where p is a prime.

1.2 Cyclic groups and generators

If there exists an element $g \in G$ with order equal to |G| then we say the group is cyclic. We say the element g generates the group and that g is a generator or primitive element of the group.
Interestingly, the group $\mathbb{Z}_p^*$ is always cyclic. This means that for some $g \in \mathbb{Z}_p^*$, we have

$$\{g^1, g^2, \ldots, g^{p\text{-}1}\} = \mathbb{Z}_p^*$$

1.3 Discrete Logarithm

The discrete logarithm problem (DLP) for $\mathbb{Z}_p^*$ is

$$\text{Given } g, b \in \mathbb{Z}_p^* \text{ then find } x \text{ such that } b \equiv g^x (\mathrm{mod} p).$$

2 One Way Functions

A one-way function is a function that is easy to compute on every input, but hard to invert given the image of a random input.

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

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

3 Hardcore Predicates

A hardcore predicate is the hardest bit of information about the input to obtain.
A function hc : $\{0,1\}^* \rightarrow \{0,1\}$ is a hard-core predicate of a function f if

1. It can be computed in polynomial time, and

2. for every probabilistic polynomial-time algorithm A, there exists a negligible function negl such that

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(f(x)) = \mathrm{hc}(x)] \leq \frac{1}{2} + \mathrm{negl}(n)$$

4 Pseudorandom Generators

4.1 What is a Pseudorandom Generator?

Pseudo Random Numbers (PRNs) are a type of random number created from a seed value. Pseudo Random Number Generators (PRGs), also known as Deterministic Random Number Generators, generate PRNs.

Let $l(.)$ be a polynomial and let G be a deterministic polynomial-time algorithm such that for any input $s \in \{0,1\}^n$, algorith G outputs a string of length $l(n)$. We say that G is a PRG if 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:

$$|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \leq \mathrm{negl}(n)$$

where r is chosen uniformly at random from $\{0,1\}^n$ and the seed s is chosen uniformly at random from $\{0,1\}^n$.

4.2 Designing secure Encryption Scheme using PRG

- Gen: on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$, output the ciphertext

$$c := G(k) \oplus m.$$

- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$, output the plaintext message

$$m := G(k) \oplus c.$$

4.3 From one-way function to single bit expansion of PRG

Let f be a one-way function and let hc be a hardcore predicate of f. Then

$$G(s) = (f(s), hc(s))$$

makes a PRG with expansion factor $l(n) = n + 1$

4.4 Single bit expansion to arbitrary expansion PRG

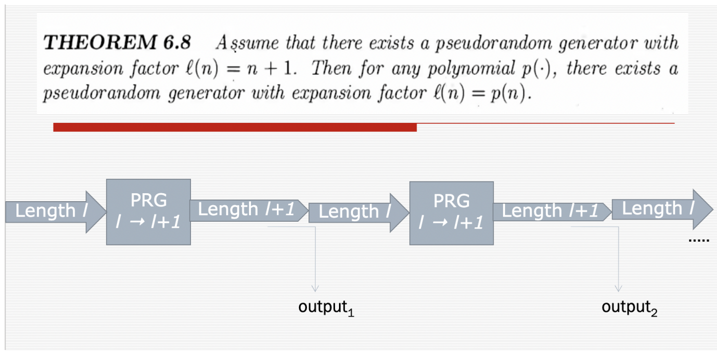

Figure 1: Single bit to Arbitrary length expansion of PRG

let us assume that a PRG with expansion factor $l(n) = n + 1$ exists. Then we do the following to create an $l(n)$ length of pseudorandom output:

- Take last bit from l+1 string for output
- Apply $l^{'}$ times to get output string of length $l^{'}$