4.3 Perfect Pseudorandom Generators

A sound definition of the concept "pseudorandom generator" is overdue. Informally we define it as an efficient algorithm that takes a "short" bitstring $s \in \mathbb{F}_2^n$ and converts it into a "long" bitstring $s \in \mathbb{F}_2^r$, compare Appendix A.2.

The terminology of complexity theory as in Appendix B of Part III allows us to give a mathematically exact (but not completey satisfying from a practical point of view) definition by considering parameter-dependent families of Boolean maps (or circuits) $G_n \colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^{r(n)}$, and analyzing their behaviour when the parameter n grows to infinity. Such an algorithm—represented by the family $(G_n)_{n\in\mathbb{N}}$ of Boolean circuits—can be efficient only if the "expanding function" $r \colon \mathbb{N} \longrightarrow \mathbb{N}$ grows at most polynomially with the parameter n, otherwise even writing down the output sequence in an efficient way is impossible. We measure the complexity in a meaningful way by the size of the circuit (or by counting the number of needed bit operations) that likewise must grow at most polynomially with n.

To make this idea more precise we consider an infinite parameter set $M \subseteq \mathbb{N}$. We assume that an instance of the generator is defined for each $m \in M$. As an example think of M as a set of Blum integers. Let $M_n = M \cap [2^{n-1} \ldots 2^n[$ be the set of n-bit integers in M.

A **pseudorandom generator with parameter set M and expansion function** r is a family $G = (G_m)_{m\in M}$ of Boolean circuits

$$G_m \colon A_m \longrightarrow \mathbb{F}_2^{r(n)} \quad \text{with } A_m \subseteq \mathbb{F}_2^n,$$

where n is the bitlength of m, such that there exists a (deterministic) polynomial family of circuits $\tilde{G} = (\tilde{G}_n)_{n\in\mathbb{N}}$, where $\tilde{G}_n$ has $2n$ deterministic input nodes, with $\tilde{G}_n(m, x) = G_m(x)$. (In other words: The pseudorandom bits are efficiently computable. In particular the function r is bounded by a polynomial in n.) A_m is called the set of seeds for the parameter m. Thus each G_m expands an n-bit sequence $x \in A_m$ to a $r(n)$-bit sequence $G_m(x) \in \mathbb{F}_2^{r(n)}$.

To see how the BBS generator fits into this definition let M be the set of Blum integers or an infinite subset of it, $A_m = \mathbb{M}_m$, and $G_m(x) = (b_1(x), \ldots, b_{r(n)}(x))$ be the corresponding BBS sequence, $b_i(x) = \mathrm{lsb}(x_i)$ where $x_0 = x$, $x_i = x_{i-1}^2 \bmod m$, for $m \in M$.

A **polynomial test** for the pseudorandom generator G is a polynomial family of (probabilistic) circuits $C = (C_n)_{n\in\mathbb{N}}$,

$$C_n : \mathbb{F}_2^n \times \mathbb{F}_2^{r(n)} \times \Omega_n \longrightarrow \mathbb{F}_2$$

over a probability space $\Omega_n \subseteq \mathbb{F}_2^{s(n)}$ where $s(n)$ is the number of probabilistic inputs of C_n. Thus the test C_n may depend on the parameter m. The probability that the test computes the value 1 for a sequence generated by G is

$$p(G, C, m) = P\{(x, \omega) \in A_m \times \Omega_n \mid C_n(m, G_m(x), \omega) = 1\}.$$

The probability that the test computes the value 1 for an arbitrary ("true random") sequence of the same length is

$$\bar{p}(C, m) = P\{(u, \omega) \in \mathbb{F}_2^{r(n)} \times \Omega_n \mid C_n(m, u, \omega) = 1\}.$$

Ideally (for a "good" generator) these two probabilities should agree approximately: the test should not be an ε-distinguisher for reasonable values of ε and for almost all parameters m. We say the pseudorandom generator G **passes the test** C if for all non-constant polynomials $h \in \mathbb{N}[X]$ the set A of $m \in M$ with

$$|p(G, C, m) - \bar{p}(C, m)| \geq \frac{1}{h(n)}$$

is sparse in M (the set of parameters m for which C is a $1/h(n)$-distinguisher).

Recall from Appendix B.7 of Part III that this means that

$$\frac{\#(A \cap M_n)}{\#M_n} \leq \frac{1}{\eta(n)} \quad \text{for almost all } n \in \mathbb{N}$$

for each non-constant polynomial $\eta \in \mathbb{N}[X]$.

The pseudorandom generator G is called **perfect** if it passes all polynomial tests. In sloppy words:

No efficient statistical test (or algorithm) is able to distinguish a bit sequence generated by G from a "true random" bit sequence.