Recommended Reading.

- Katz–Lindell §6.4.

We will prove:

**Theorem 1** If one-way permutations exist, then pseudorandom generators exist (for any expansion function $\ell(n) = \text{poly}(n)$).

The construction consists of two stages:

- One-way permutations + hardcore bit $\Rightarrow$ PRGs that stretch by 1 bit
- PRGs with 1-bit stretch $\Rightarrow$ PRGs with “arbitrary” stretch.

## 1 Hardcore Bits $\Rightarrow$ PRGs with 1-bit Stretch

**Theorem 2** If $f$ is a one-way permutation with hardcore bit $b$, then $G(s) = f(s)b(s)$ is a pseudorandom generator.

**Proof:**

1. Suppose there is a PPT $D$ that distinguishes between $G(S) = f(S)b(S)$ and $U_{n+1} = f(S)R$ with nonnegligible advantage $\varepsilon$ (where $S \leftarrow \{0,1\}^n$ and $R \leftarrow \{0,1\}$).
2. Then $D$ distinguishes between $Y_0 = f(S)b(S)$ and $Y_1 = f(S)b(S)$ with advantage $2\varepsilon$.
3. We can construct a PPT $A$ that predicts $C$ from $Y_C = f(S) \circ (b(S) \oplus C)$, where $C \leftarrow \{0,1\}$, with probability at least $1/2 + \varepsilon$.
4. $B(f(S)) = A(f(S)C') \oplus C'$, where $C' \leftarrow \{0,1\}$, predicts $b(S)$ with probability at least $1/2 + \varepsilon$. This contradicts the definition of hardcore bit.

## 2 Increasing the Expansion

First attempt: run $G$ with many independent seeds.

**Theorem 3** Let $G : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ be a PRG. Then $G'(s_1s_2\cdots s_\ell) = G(s_1)G(s_2)\cdots G(s_\ell)$ is a PRG for any $\ell \leq \text{poly}(n)$. 

Proof: "Hybrid technique". For \( i = 0, \ldots, \ell \), define the hybrid \( H_i = R_1 R_2 \cdots R_i G(S_{i+1}) \cdots G(S_\ell) \), where \( R_j \overset{\text{R}}{\leftarrow} \{0,1\}^{n+1} \) and \( S_j \overset{\text{R}}{\leftarrow} \{0,1\}^{n} \). Then \( H_0 \equiv G'(U_{\ell n}) \) and \( H_\ell \equiv U_{\ell n+\ell} \).

Suppose that \( G' \) is not a PRG: there exists a PPT \( D \) such that:

\[
\Pr[D(G'(U_{\ell n})) = 1] - \Pr[D(U_\ell) = 1] > \varepsilon
\]

where \( \varepsilon \) is nonnegligible. This inequality can be rewritten using the hybrids \( H_i \):

\[
\sum_{i=0}^{\ell-1} (\Pr[D(H_i) = 1] - \Pr[D(H_{i+1}) = 1]) > \varepsilon,
\]

so there exists an \( i \) such that

\[
\Pr[D(H_i) = 1] - \Pr[D(H_{i+1}) = 1] > \frac{\varepsilon}{\ell}.
\]

Then the PPT \( D'(x) = D(R_1 \cdots R_i x G(S_{i+2}) \cdots G(S_\ell)) \) distinguishes \( G(S_{i+1}) \equiv G(U_n) \) from \( R_{i+1} \equiv U_{n+1} \) with advantage \( \varepsilon/\ell \). \( \implies \)

Better approach: composition.

**Theorem 4** Let \( G : \{0,1\}^n \rightarrow \{0,1\}^{n+1} \) be a PRG. Define \( G_\ell(s_0) = b_1 b_2 \cdots b_\ell \), where \( s_{i+1} b_{i+1} \overset{\text{def}}{=} G(s_i) \) for \( i = 0, \ldots, \ell - 1 \). Then, for any \( \ell \leq \text{poly}(n) \), \( G_\ell \) is a PRG with expansion \( \ell \).

**Proof:** Intuition: \( G(s_0) = (s_1, b_1) \) looks random & independent, so \( (G(s_1), b_1) = (s_2, b_2, b_1) \) looks random & independent, etc. To formalize this, we will use the hybrid technique. For \( i = 0, \ldots, \ell \), define \( H_i = U_i \circ G_{\ell-i}(U_n) \). Then \( H_0 = G_\ell(U_n) \), \( H_\ell = U_\ell \).

As above, if \( G_\ell \) is not a PRG, then there exists a PPT \( D \) such that

\[
\Pr[D(H_i) = 1] - \Pr[D(H_{i+1}) = 1] > \frac{\varepsilon}{\ell},
\]

where \( \varepsilon \) is nonnegligible.

Define the PPT \( D'(y) \):

1. Write \( y = s_{i+1} b_{i+1} \) where \( |s_{i+1}| = n \).
2. Choose \( b_1, \ldots, b_\ell \overset{\text{R}}{\leftarrow} \{0,1\} \).
3. Let \( b_{i+2} \cdots b_\ell = G_{\ell-i-1}(s_{i+1}) \).
4. Run \( D(b_1 \cdots b_\ell) \)

If \( y \leftarrow G(U_n) \), then \( D \) is fed with \( b_1 \cdots b_\ell \leftarrow H_i \).
If \( y \leftarrow U_{n+1} \), then \( D \) is fed with \( b_1 \cdots b_\ell \leftarrow H_{i+1} \).
Thus,

\[
\Pr[D'(G(U_n)) = 1] - \Pr[D'(U_{n+1}) = 1] > \frac{\varepsilon}{\ell}
\]

\( \varepsilon \) is nonnegligible and \( \ell \) is a polynomial so \( \frac{\varepsilon}{\ell} \) is nonnegligible, contradicting the assumption that \( G \) is a pseudorandom generator. \( \blacksquare \)
Generator obtained from above two theorems

If \( f \) is a one-way permutation with hardcore bit \( b \), \( G(x) = b(x)b(f(x))b(f(f(x))) \cdots b(f^\ell(x)) \).

- The bits can be computed on-line, if we remember the current value of \( s_i = f^i(s_0) \). To output a new bit, we output \( b(s_i) \) and update \( s_{i+1} \leftarrow f(s_i) \).
- The construction does not depend on \( \ell \) : the stretch doesn’t have to be determined in advance. (Note that the security degrades linearly with the number of bits produced, i.e. the adversary’s advantage increases)
- This construction also works for collections of one-way permutations.

\[
G(r_1, r_2) = b_{key}(x)b_{key}(f_{key}(x)) \cdots b_{key}(f^\ell_{key}(x))
\]

where \( r_1 \) are the coin tosses used to select \( \text{key} \leftarrow^R G(1^n) \) and \( r_2 \) are the coin tosses to sample \( x \leftarrow^R D_{key} \). The proofs are similar to the proofs above with the modification that we give the key key to the adversary since it has to be able to evaluate the function \( f_{key} \).

Concrete Instantiations

1. RSA:
   - Use the seed to pick a function from the family, i.e. pick random \( n \)-bit primes \( p, q \) \( (N = pq) \), \( e \leftarrow \mathbb{Z}_{\phi(N)}^* \), \( x \leftarrow^R \mathbb{Z}_N^* \).
   - Output: \( \text{lsb}(x), \text{lsb}(x^e \mod N), \text{lsb}(x^{e^2} \mod N), \text{lsb}(x^{e^3} \mod N), \ldots \)

2. Rabin:
   - Use the seed to choose \( p \equiv q \equiv 3 \pmod{4} \) (we need one-way permutations) and \( x \leftarrow^R \mathbb{Z}_N^* \).
   - Output: \( \text{lsb}(x^2 \mod N), \text{lsb}(x^{2^2} \mod N), \text{lsb}(x^{2^3} \mod N), \ldots \)
   - If the Factoring Assumption holds, the above construction is a pseudorandom generator.

3. Modular Exponentiation:
   - Use the seed to generate \( (p, g, x) \).
   - Output: \( \text{half}_{p-1}(x), \text{half}_{p-1}(g^x \mod p), \text{half}_{p-1}(g^{g^x \mod p} \mod p), \ldots \)

4. All of the above secure if output \( O(\log n) \) bits per iteration. Unproven (but conjectured) if output \( n/2 \) bits per iteration.