How much randomness do you need to generate $N = pq$?

- To generate a random $k$-bit prime you need $\approx k^2$ secret bits
- Hence, megabits of secret randomness to get an RSA key (expensive!)
- Pseudorandomness is good, but how do you get the seed?
  (e.g., Blum-Blum-Shub PRG needs $N$ as part of seed)

**Result 1:** Any OWF with $k$-bit output needs only a $k$-bit input
(RSA keys from short secrets)

**Result 2:** If the OWF is regular, get a PRG with a short seed
(BBS PRG with a short seed)

**New approach**
- Expand input using 2-wise independent hashing
- Replace secret randomness with public randomness

**Research Impact**
- Replaces ad hoc key gen tools with a provable simple solution
- First linear-seed-length PRG from nonpermutations

**OWF Result:**
if $h: k \text{ bits} \rightarrow n \text{ bits}$ is 2-wise indep function and $f: k \text{ bits} \rightarrow n \text{ bits}$ is one-way, then $f(h(x))$ is one-way even given the randomness used to choose $h$.

**Proof technique:** compare $(h, f(h(x)))$ to $(h, f(y))$. These are not statistically close, but probabilities of events are polynomially related (“domination”).

**Optimality:**
we prove that at least $n$ total input bits and $k$ secret input bits are necessary (assuming black-box reductions).

**PRG Result:**
if $f$ is regular (every output is equally likely), then randomized iteration of $f(h(x))$ gives a PRG with seed length $2n + O(k \log k)$ bits, of which only $k$ are secret. Seed length is linear if $k = O(n/\log n)$.