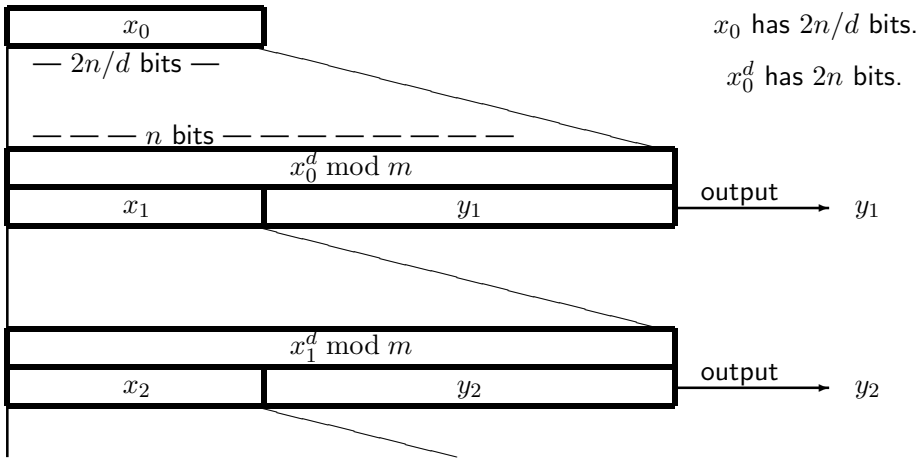

Figure 4.4: Micali-Schnorr generator

4.7 The Micali-Schnorr Generator

Micali and Schnorr proposed a pseudorandom generator that is a descendent of the RSA generator. Fix an odd number $d \geq 3$. The parameter set is the set of all products m of two primes p and q whose bit lengths differ by at most 1, and such that d is coprime with $(p-1)(q-1)$. For an n-bit number m let $h(n)$ be an integer $\approx \frac{2n}{d}$. Then the d-th power of an $h(n)$-bit number is (approximately) a $2n$-bit number.

In the i-th step calculate $z_i = x_{i-1}^d \bmod m$. Take the first $h(n)$ bits as the new state x_i, that is $x_i = \lfloor z_i / 2^{n-h(n)} \rfloor$, and output the remaining bits, that is $y_i = z_i \bmod 2^{n-h(n)}$. Thus the bits of the result z_i are partitioned into two *disjoint* parts: the new state x_i, and the output y_i. Figure 4.4 illustrates this scheme.

But why may we hope that this pseudorandom generator is perfect? This depends on the hypothesis: There is no efficient test that distinguishes the uniform distribution on $\{1, \ldots, m-1\}$ from the distribution of $x^d \bmod m$ for uniformly distributed $x \in \{1, \ldots, 2^{h(n)}\}$. If this hypothesis is true, then the Micali-Schnorr generator is perfect. This argument seems tautologic, but heuristic considerations show a relation with the security of RSA and with factorization. Anyway we have to concede that this "proof of security" seems considerably more airy then that for BBS.

How fast do the pseudorandom bits tumble out of the machine? As elementary operations we again count the multiplication of two 64-bit numbers, and the division of a 128-bit number by a 64-bit number with 64-bit quotient. We multiply and divide by the classical algorithms. Thus the product of s (64-bit) words and t words costs st elementary operations. The cost of

division is the same as the cost of the product of divisor and quotient.

The concrete recommendation by the inventors is: $d = 7$, $n = 512$. (Today we would choose a larger n.) The output of each step consists of 384 bits, withholding 128 bits as the new state. The binary power algorithm for a 128-bit number x with exponent 7 costs several elementary operations:

- x has 128 bits, hence 2 words.
- x^2 has 256 bits, hence 4 words, and costs $2 \cdot 2 = 4$ elementary operations.
- x^3 has 384 bits, hence 6 words, and costs $2 \cdot 4 = 8$ elementary operations.
- x^4 has 512 bits, hence 8 words, and costs $4 \cdot 4 = 16$ elementary operations.
- x^7 has 896 bits, hence 14 words, and costs $6 \cdot 8 = 48$ elementary operations.
- $x^7 \bmod m$ has ≤ 512 bits, and likewise costs $6 \cdot 8 = 48$ elementary operations.

This makes a total of 124 elementary operations; among them only one reduction $\bmod m$ (for x^7). Our reward consists of 384 pseudorandom bits. Thus we get about 3 bits per elementary operation, or, by the assumptions in Section 4.6, about 6 milliards bits per second. Compared with the BBS generator this amounts to a factor of about 1000.

Parallelization increases the speed virtually without limit: The Micali-Schnorr generator allows complete parallelization. Thus distributing the work among k CPUs brings a profit by the factor k since the CPUs can work indepedently of each other without need of communication.