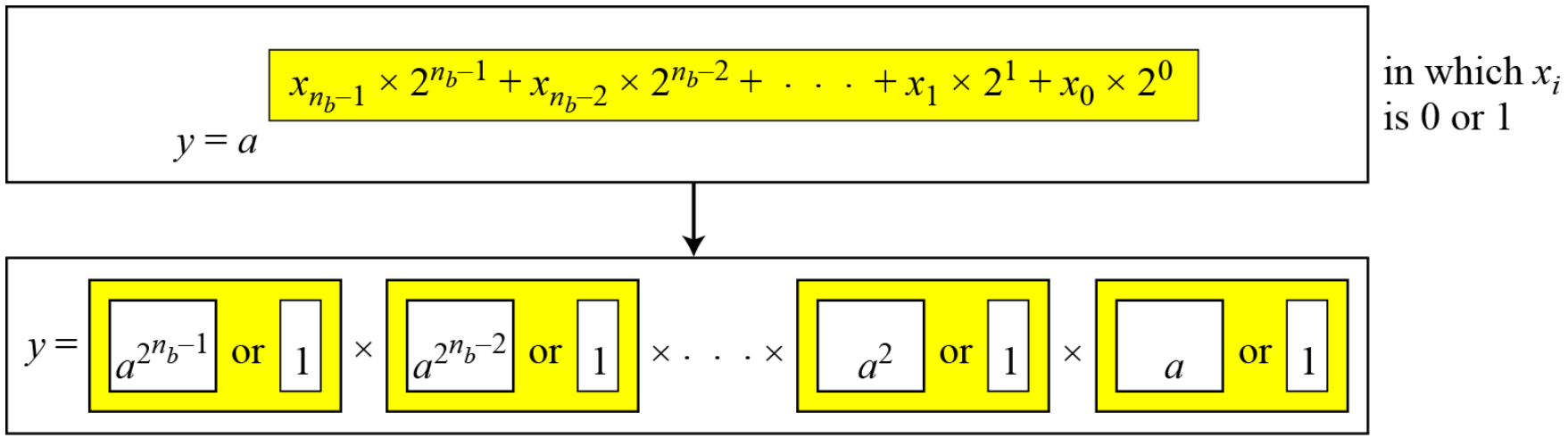
EXPONENTIATION

Exponentiation: $y = a^x$ $\rightarrow$ **Logarithm:** $x = \log_a y$

9.6.1 Exponentiation

Fast Exponentiation

Figure 9.6 *The idea behind the square-and-multiply method*



Example:

$$y = a^9 = a^{1001_2} = a^8 \times 1 \times 1 \times a$$

9.6.1 Continued

Algorithm 9.7 *Pseudocode for square-and-multiply algorithm*

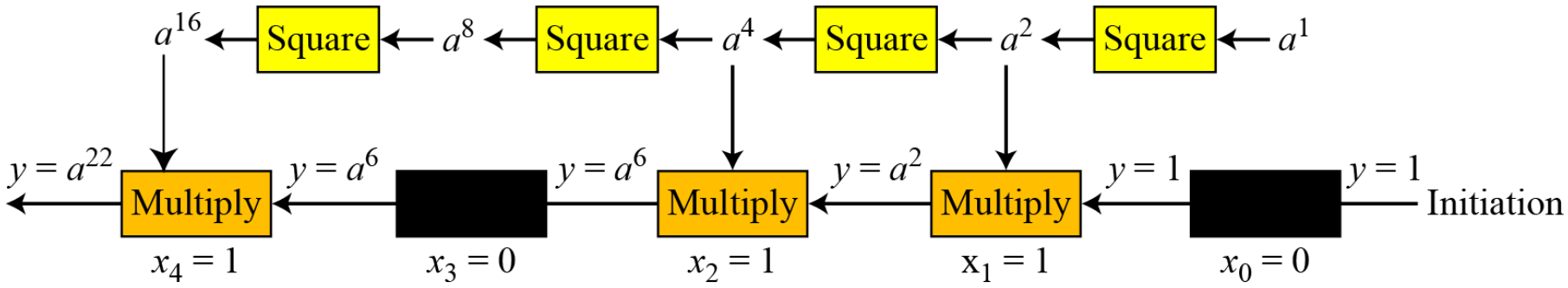
```
Square_and_Multiply (a, x, n)
{
    y ← 1
    for (i ← 0 to n_b − 1)                  // n_b is the number of bits in x
    {
        if (x_i = 1)    y ← a × y mod n     // multiply only if the bit is 1

        a ← a^2 mod n                       // squaring is not needed in the last iteration
    }
    return y
}
```

9.6.1 Continued

Example 9.45

Figure 9.7 shows the process for calculating $y = a^x$ using the Algorithm 9.7 (for simplicity, the modulus is not shown). In this case, $x = 22 = (10110)2$ in binary. The exponent has five bits.

Figure 9.7 *Demonstration of calculation of a^{22} using square-and-multiply method*



9.6.1 Continued

Table 9.3 *Calculation of 17^{22} mod 21*

i	x_i	*Multiplication (Initialization: $y = 1$)*	*Squaring (Initialization: $a = 17$)*
0	0	→	$a = 17^2 \bmod 21 = 16$
1	1	$y = 1 \times 16 \bmod 21 = 16$ →	$a = 16^2 \bmod 21 = 4$
2	1	$y = 16 \times 4 \bmod 21 = 1$ →	$a = 4^2 \bmod 21 = 16$
3	0	→	$a = 16^2 \bmod 21 = 4$
4	1	$y = 1 \times 4 \bmod 21 = 4$ →	

Note

The bit-operation complexity of the fast exponential algorithm is polynomial.