Exercises, chapter 4, solutions

---

1. The algorithm computes the value of $g^x \bmod p$ using the formula

$$g^x \bmod p = (g^{2^{n-1}})^{b_{n-1}} \times (g^{2^{n-2}})^{b_{n-2}} \times \cdots \times (g^{2^1})^{b_1} \times (g^{2^0})^{b_0} \bmod p,$$

where $b_{n-1}, b_{n-2}, \ldots b_1, b_0$ are the bits in the binary representation of x. Each iteration of the **while**-loop takes care of one $(g^{2^i})^{b_i}$-factor in the formula, from the right to the left. More precisely, for $i \in \{0, 1, \ldots, n-1\}$, iteration i of the **while**-loop updates the product r that has been computed so far according to one of two cases by doing the following:

- On line 5, it checks whether the bit b_i is equal to 0 or 1. The rightmost bit of the variable d is equal to b_i, so the algorithm determines the value of b_i by calculating $d \bmod 2$. After that:
  - If $b_i = 1$ (which means that $(g^{2^i})^{b_i} = g^{2^i}$) then on line 6, the algorithm will multiply r by the value g^{2^i} (currently stored in c) and apply the (mod p)-operation.
  - On the other hand, if $b_i = 0$ (which means that $(g^{2^i})^{b_i} = 1$) then line 6 will be skipped and r left unchanged.
- Line 7 updates the variable d for the next iteration by making the rightmost bit of d become b_{i+1}. This is achieved by dividing d by 2 and rounding the result down to the nearest integer, which has the effect of shifting all of the bits of d one position to the right.
- Line 8 updates the variable c for the next iteration by setting it to $(g^{2^i} \times g^{2^i}) \bmod p = g^{2 \cdot 2^i} \bmod p = g^{2^{i+1}} \bmod p$.

2. (a) $a \cdot b = 9 \cdot 4 = 36 = 1 \cdot 32 + 0 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1$, so $a \cdot b$ is 100100 in binary.

   (b) The shared secret key will be $g^{ab} \bmod p = 7^{9 \cdot 4} \bmod 23$. We can compute this value by applying the algorithm for modular exponentiation by repeated squaring from question 1. with $g = 7$, $x = 9 \cdot 4$, and $p = 23$ as shown in the table below. Note that the initial value of d in the table is equal to x, which we know from part (a) is 100100 when written in binary.

i	$c = 7^{2^i} \bmod 23$	r	d
0	7	1	100100
1	$(7 \cdot 7) \bmod 23 = 3$	1	10010
2	$(3 \cdot 3) \bmod 23 = 9$	1	1001
3	$(9 \cdot 9) \bmod 23 = 12$	$(1 \cdot 9) \bmod 23 = 9$	100
4	$(12 \cdot 12) \bmod 23 = 6$	9	10
5	$(6 \cdot 6) \bmod 23 = 13$	9	1
		$(9 \cdot 13) \bmod 23 = 2$	

(The answer is the value of r in the last row, i.e., 2.)

   Thus, the shared secret key will be: $g^{ab} \bmod p = 7^{9 \cdot 4} \bmod 23 = 2$

   Remark: To compute the value of $7^{9 \cdot 4} \bmod 23$ without manually running the algorithm for modular exponentiation by repeated squaring, one can use a calculator or mathematical software. (For example, type "*modp*($7^{9 \cdot 4}$, 23);" in Maple or use the Python function `pow` with three arguments.)

(c) Bob computes and transmits $B = g^b \bmod p = 7^4 \bmod 23 = 9$ to Alice. After receiving B, Alice computes $B^a \bmod p = 9^9 \bmod 23 = 2$.

For completeness, the calculations of $7^4 \bmod 23$ and $9^9 \bmod 23$ using the same method as in part (b) are also given here. 4 is 100 in binary and 9 is 1001 in binary.

i	$c = 7^{2^i} \bmod 23$	r	d
0	7	1	$10\underline{0}$
1	$(7 \cdot 7) \bmod 23 = 3$	1	$1\underline{0}$
2	$(3 \cdot 3) \bmod 23 = 9$	1	$\underline{1}$
		$(1 \cdot 9) \bmod 23 = 9$	

i	$c = 9^{2^i} \bmod 23$	r	d
0	9	1	$100\underline{1}$
1	$(9 \cdot 9) \bmod 23 = 12$	$(1 \cdot 9) \bmod 23 = 9$	$10\underline{0}$
2	$(12 \cdot 12) \bmod 23 = 6$	9	$1\underline{0}$
3	$(6 \cdot 6) \bmod 23 = 13$	9	$\underline{1}$
		$(9 \cdot 13) \bmod 23 = 2$	

3. The decrypted message is: `Nice work!!`

   Here is one of many available implementations of AES:
   `https://encode-decode.com/aes192-encrypt-online/`