**Solutions for week 3, Cryptography Course - TDA 352/DIT 250**

The problems this week are partly simple exercises in using primitives using small numbers, partly old exam problems, intended to test these aspects (for the second item, mostly improper use of primitives).

# Easy

1. (a) Let us first recall how the Extended Euclidean Algorithm works. The aim of the Euclidean Algorithm is to find the greatest commod divisor (GCD) of two numbers. This is done as follows: let $a, b$ be two numbers (positive integers), then it is either the case that $a = b$ or $a > b$ (in case $b$ is bigger than $a$ we can simply swap the two numbers). If $a = b$ then it is obvious that the GCD$(a, b)$=$a$=$b$. The other case is the non-trivial one (which is indeed what the exercise requires with $a = 25$ and $b = 42$), and we can solve it in this way:

   i. Find a positive integer $q_1 \geq 1$ such that $a = bq_1 + r_1$, for some integer $r_1$ smaller than $b$, i.e. $0 \leq r_1 < b$.
   ii. Find a non-negative integer $q_2 \geq 0$ such that $b = r_1q_2 + r_2$, for some integer $r_2$ satisfying $0 \leq r_2 < r_1$.
   iii. Find a non-negative integer $q_3 \geq 0$ such that $r_1 = r_2q_3 + r_3$, for some integer $r_3$ satisfying $0 \leq r_3 < r_2$.
   iv. Proceed in this way (the general step is $r_i = r_{i+1}q_{i+2} + r_{i+2}$) until the last remainder is null, i.e. $r_{i+2} = 0$. Then it means that the previous reminder, $r_{i+1}$, is the greatest common divisor of $a$ and $b$, i.e. $r_{i+1} = GCD(a, b)$.

   If we proceed according to the above recipe, we will get:

$$\begin{array}{rcrl||rcrl} 42 & = & 25(1) & +17 & a & = & b(q_1) & +r_1 \\ 25 & = & 17(1) & +8 & b & = & r_1(q_2) & +r_3 \\ 17 & = & 8(2) & +1 & r_1 & = & r_2(q_3) & r_3 \\ 8 & = & 1(8) & 0 & r_2 & = & r_3(q_4) & r_4 \end{array}$$

   The GCD of 42 and 25 is the reminder of the second-last row: $r_3 = 1$. Therefore, the two numbers are relatively prime.

   In order to find $x$ and $y$ such that $42x+25y = 1$ we need to use the Extended Euclidean Algorithm. We start from the second-last row in the above table, and read the equations "backwards" in order to obtain $r_3$ (the GCD) written in terms of $a$ and $b$ (the two given numbers). We proceed as shown in the next table:

$$\begin{array}{rcrcl} 17 & = & (8)2 & +1 & \Longrightarrow \quad 1 = 17 + 2(-8) \\ 25 & +17(-1) & = & 8 & \Longrightarrow \quad 1 = 17 + 2(-(25-17)) = 25(-2) + 3(17) \\ 42 & +25(-1) & = & 17 & \Longrightarrow \quad 1 = 25(-2) + 3(42-25) \\ & & & & \quad\;\; 1 = = 42(3) + 25(-5) \end{array}$$

   Thus, $x = 3$ and $y = -5$.

   (b) For any integer $t$, we can add and subtract $42 \cdot 25 \cdot t$ to the left hand side, giving

$$42 \cdot (3 + 25t) + 25 \cdot (-5 + 42t) = 1.$$

   (c) From $42 \cdot 3 - 25 \cdot 5 = 1$ we immediately get that $25 \cdot (-5) = 1$ in $\mathbb{Z}_{42}^*$, so $25^{-1} = -5 = 37 \mod 42$.

2. Recall that Fermat's theorem states that for a given prime $p$ relatively prime to $a$, $a^{p-1} = 1 \pmod p$.

   (a)

$$\begin{aligned} 3^{102} &= 3^{100} \cdot 3^2 \\ &= (3^{10})^{10} \cdot 3^2 \\ &= (1)^{10} \cdot 9 = 9 \pmod{11} \end{aligned}$$

(b)

$$\begin{aligned} 3^{503} &= 3^{500} \cdot 3^3 \\ &= (3^{10})^{50} \cdot 3^3 \\ &= (1)^{50} \cdot 27 = 27 = 5 \pmod{11} \end{aligned}$$

3. (a) We have $\Phi(N) = (11-1)(17-1) = 160$. To compute $d$ we use the Extended Euclidean Algorithm between $\Phi(N) = 160$ and the encryption exponent $e = 3$:

$$\begin{array}{rcrl} 160 & = & 3(53) & +1 \\ 3 & = & 1(3) & +0 \end{array}$$

From the second-last row (which is actually the first one!) we see that $1 = 160 + 3(-53)$. If we read the equation in $\mathbb{Z}_{160}$ we directly get that the inverse of $e = 3 \mod 160$ is $-53 = d$. For simplicity we take the positive representative of the equivalence class of $-53$ in $\mathbb{Z}_{160}$, that is $d = 107$ $(= -53)$ in $\mathbb{Z}_{160}$.

(b) Here, $M = d$, which is just a coincidence (extremely unlikely for realistic numbers!).
We need to compute $107^3$ in $\mathbb{Z}_{187}$. Two modular multiplications give $107 \cdot 107 = 42$, $42 \cdot 107 = 6$. So the encrypted message is $c = 6$.

For decryption we need to compute $6^{107}$. We use the repeated squaring algorithm:

| $n$ | $2^n$ | $6^{2^n}$ |
|---|---|---|
| 0 | 1 | 6 |
| 1 | 2 | 36 |
| 2 | 4 | 174 |
| 3 | 8 | 169 |
| 4 | 16 | 137 |
| 5 | 32 | 69 |
| 6 | 64 | 86 |

We note that $107 = 64+32+8+2+1$ and compute $86 \cdot 69 \cdot 169 \cdot 36 \cdot 6 = 107$. The multiplications here are of course in $\mathbb{Z}^*_{187}$, so one should reduce each product modulo 187 to keep numbers small.

(c) The encryption is $110^3$ in $\mathbb{Z}^*_{187}$, i.e.

$$(((110 \cdot 110) \bmod 187) \cdot 110) \bmod 187 = (132 \cdot 110) \bmod 187 = 121.$$

Decryption is $121^{107}$ in $\mathbb{Z}^*_{187}$. To compute this we use the modular exponentiation algorithm:

| $n$ | $2^n$ | $121^{2^n}$ |
|---|---|---|
| 0 | 1 | 121 |
| 1 | 2 | 55 |
| 2 | 4 | 33 |
| 3 | 8 | 154 |
| 4 | 16 | 154 |
| 5 | 32 | 154 |
| 6 | 64 | 154 |

As in (a), we multiply results corresponding to 1 bits in 107, giving

$$121 \cdot 55 \cdot 154 \cdot 154 \cdot 154 = 110.$$

**Note:** A surprising fact that can be noted here is that the number 154 repeats in the column of repeated squares. Let us analyze the phenomenon a bit.
We have, in fact, that $154^2 = 154$ in $\mathbb{Z}^*_{187}$. This could not happen in $\mathbb{Z}^*_p$ for a prime $p$, because there we have the cancellation law that allows us to cancel an $x$ in $x^2 = x$, giving $x = 1$. But 187 is not a prime and, indeed, $\gcd(110, 187) = 11 = p$, i.e. in this example the message is a multiple of one of the two primes. (Estimate the probability for this to happen in case $p$ and $q$ are 500-bit primes!) When this happens, the encrypted value will also be a multiple of $p$. We leave further analysis of this to the interested reader.

4. The problem is solved by computing $\{g^n | n \in \mathbb{Z}_{17}^*\}$ for all possible $g$, i.e. $g = 1, 2, \ldots 16$ and check for which $g$ the result is all of $\mathbb{Z}_{17}$.

   Doing this, one finds the generators 3, 5, 6, 7, 10, 11, 12 and 14.

   For subgroups, the computations show the following:

| Subgroup | Generators |
|---|---|
| $\{1\}$ | 1 |
| $\{1, 16\}$ | 16 |
| $\{1, 4, 13, 16\}$ | $4, 13$ |
| $\{1, 2, 4, 8, 9, 13, 15, 16\}$ | $2, 8, 9, 15$ |
| $\mathbb{Z}_{17}^*$ | $3, 5, 6, 7, 10, 11, 12, 14$ |

5. Since $15 = 3 \cdot 5$, we have that

$$\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$$

   (only numbers $k$ with $1 \leq k \leq 14$ and $\gcd(k, 15) = 1$ are elements of $\mathbb{Z}_{15}^*$). The order of the group is, of course, $\Phi(15) = (3-1)(5-1) = 8$.

   We can check each of the elements to see what subgroup they generate and will find

| Subgroup | Generators |
|---|---|
| $\{1\}$ | 1 |
| $\{1, 4\}$ | 4 |
| $\{1, 11\}$ | 11 |
| $\{1, 14\}$ | 14 |
| $\{1, 2, 4, 8\}$ | $2, 8$ |
| $\{1, 4, 7, 13\}$ | $7, 13$ |
| $\mathbb{Z}_{15}^*$ | $--$ |

   In particular, we see that there is no generator for the whole group (the group is not cyclic).

6. $\mathbb{Z}_{83}^*$ has $82 = 2 \cdot 41$ elements, so the order of any element is either 1, 2, 41 or 82 (in the last case the element is a generator). Obviously, $3^2 = 9$, so 3 does not have order 1 or 2. So, if 3 is not a generator, its order must be 41, i.e. $3^{41} = 1$, where the calculation is done in $\mathbb{Z}_{83}^*$. We check this by computation:

   By successive squaring we get $3^4 = 9^2 = 81 = -2$, $3^8 = (-2)^2 = 4$, $3^{16} = 4^2 = 16$ and $3^{32} = 16^2 = 256 = 7$. Combining, we get

$$3^{41} = 3^{32} \cdot 3^8 \cdot 3^1 = 7 \cdot 4 \cdot 3 = 84 = 1.$$

   We have thus confirmed that 3 is, indeed, not a generator of $\mathbb{Z}_{83}^*$ Instead, we have shown that the subgroup generated by 3 has order 41.

---

# Medium

---

7. Recall that Euler's theorem states that for a given positive integer $N$ relatively prime to $a$, $a^{\varphi(N)} = 1 \pmod{N}$. To find the least significant decimal digit, we work modulo 10. We have $\varphi(10) = 4$, so that

$$3^{100} = (3^4)^{25} = (1)^{25} = 1 \pmod{10}$$

   Thus, the last digit of $3^{100}$ is 1.

8. We wish to solve the following system of linear congruences

$$x = 1 \pmod{3}$$
$$x = 4 \pmod{7}$$

   From the Chinese Remainder Theorem, we know that the solution is given by $x = (1)(s)(7) + (4)(t)(3) \pmod{3 \cdot 7}$, where $s, t$ are the Bézout's coefficients of 7 and 3 respectively. To find these coefficients, we apply the extended Euclidean algorithm:

| r | 7s | + | 3t | q |
|---|---|---|---|---|
| 7 | 1 | | 0 | |
| 3 | 0 | | 1 | 2 |
| 1 | 1 | | -2 | 3 |
| 0 | | | | |

From the table, we get that $s = 1, t = -2$, so that $x = (1)(1)(7) + (4)(-2)(3) = 7 - 24 = -17 = 4 \pmod{21}$. Therefore, $x = \mathbf{4}$.

9. Recall that
$$\Phi(N) = (p-1)(q-1) = pq - p - q + 1 = N - p - q + 1.$$
So if the Adversary knows $\Phi(N)$, he also knows $p + q = N + 1 - \Phi(N)$.

But if you know both $p \cdot q = N$ and $p + q = a$ it is easy to compute $p$ and $q$: using $q = N/p$, you know $p + N/p = a$, which gives the ordinary second-degree equation $p^2 - ap + N = 0$ to solve for $p$.

10. (a) It is required that $\gcd(e, \Phi(N)) = 1$ for the RSA system to work properly. But $\Phi(N) = (p-1)(q-1)$ is an even number, so $\gcd(2, \Phi(N)) = 2$. In particular, it would not be possible to choose $d$ with $ed = 1 \bmod \Phi(N)$.

    (b) The general argument against double encryption is that it is subject to the meet-in-the-middle attack, which has time complexity similar to that of a single brute force attack. In the particular case of RSA encryption, double encryption is also meaningless, since the double encryption is equivalent to a single RSA encryption with public key $e_1 e_2$ and private key $d_1 d_2$. To verify this, see that
$$(m^{e_1} \bmod N)^{e_2} \bmod N = m^{e_1 e_2} \bmod N.$$

    (c) Counter mode is unusable, since here decryption is the same as encryption and anyone can encrypt messages, using the receiver's public key. Hence anyone intercepting a ciphertext can decrypt it. CBC mode does not have this disadvantage, since it uses the decryption function.

11. Let us consider the group $\mathbb{Z}_{25}$.

    (a) The invertible elements in $\mathbb{Z}_{25}$ are all the elements $a \in \mathbb{Z}_{25}$ such that $\gcd(a, 25) = 1$.
    You can test every element $a \in \{0, \cdots, 24\}$ or you can do in a different way:
    we start from the fact that $25 = 5^2$ and so $5|25$ and then consider
$$\gcd(a, 25) = \begin{cases} 25 & \text{if } a = 25 \cdot k \text{ for some } k \in \mathbb{Z} \\ 5 & \text{if } a = 5 \cdot k \text{ for some } k \in \mathbb{Z} \text{ but } \gcd(k, 5) \equiv 1 \\ 1 & \text{otherwise} \end{cases}$$
    In this way, we can list all the numbers from $\{0, \cdots, 24\}$ and then we can group them with respect to the gcd:
    - 0 is not invertible, i.e., there does not exist an element $b$ such that $0b \equiv 1 \pmod{25}$
    - all the multiples of 5 (i.e. $5, 10, 15, 20$) will have $\gcd(a, 25) = 5$ since they are divisible by a prime factor of 25
    - $\{\}$ will have $\gcd(a, 25) = 25$ since we does not have any element $a \geq 25$
    - all other elements will have $\gcd(a, 25) = 1$, and they are
$$\{1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24\} = \mathbb{Z}_{25}^*$$

    (b) To check if $\langle 2 \rangle = \mathbb{Z}_{25}^*$, we should check if
$$\langle 2 \rangle = \{2^i \pmod{25} : i \in \{1, \cdots, \phi(25)\}\} = \mathbb{Z}_{25}^*$$
    To prove it, we can do either of the following two methods:
    i. compute $\{2^i \pmod{25} : i \in \{1, \cdots, \phi(25)\}\}$ and check if it is $\mathbb{Z}_{25}^*$.
    ii. check properties on the order of the subgroup generated by 2.

    We will describe the second method as it will be exploited further in the next question.
    From Euler theorem, we have $2^{\phi(25)} \equiv 1 \pmod{25}$ and if 2 is a generator, we need to have $2^i \neq 1 \pmod{25}$ for all $i \in \{1, \cdots, (\phi(20) - 1)\}$.

    The additional information[1] that we have is

[1] To be precise, it is really common theorem in group theory.

Let $G$ be a group and $H$ a subgroup of $G$.
*The order of a subgroup $H$ divides the order of the group $G$.*

From this fact, we have that $\text{ord}(\langle 2\rangle) | \phi(25) = 20 = 2^2 \cdot 5 = \text{ord}\,(\mathbb{Z}_{25}^*)$.

To prove that 2 is a generator, we just need to check that for every number $p$ such that $p|\phi(25) = 20$ and $p \neq 1$, we have

$$2^{\frac{\phi(25)}{p}} \neq 1 \pmod{25}$$

This computation is a proof because computing $2^{\frac{\phi(25)}{p}}$ and checking if it is $\equiv 1 \pmod{()25}$ means that we are checking if $\langle 2\rangle$ has order $p$ (if the $\equiv$ holds).

$$2^{\frac{\phi(25)}{2}} = 2^{\frac{20}{2}} = 2^{10} = 1024 \equiv 24 \pmod{25} \neq 1 \pmod{25}$$

$$2^{\frac{\phi(25)}{5}} = 2^{\frac{20}{5}} = 2^4 \equiv 16 \pmod{25} \neq 1 \pmod{25}$$

It is not necessary to check $p = 4 = 2^2$ since

$$24 \pmod{25} \equiv 2^{\frac{20}{2}} = 2^{\frac{20}{2}\cdot\frac{2}{2}} = 2^{\left(\frac{20}{4}\right)\cdot 2}$$

and so we have that $2^{\frac{20}{4}}$ cannot be 1 because otherwise we have $1^2 \equiv 24 \pmod{25}$ which is impossible.

(c) Using the subgroup order theorem, we can easily generate the subgroup of order $i$ by computing

$$\langle 2^{\frac{\phi(25)}{i}} \rangle = \text{ subgroup of } \mathbb{Z}_{25}^* \text{ of order } i$$

For this reason we have:

- $i = 1 : \langle 2^{20}\rangle = \langle 1\rangle$ as the subgroup of order 1
- $i = 2 : \langle 2^{10}\rangle = \langle 24\rangle$ as the subgroup of order 2
- $i = 4 : \langle 2^{5}\rangle = \langle 16\rangle$ as the subgroup of order 4
- $i = 5 : \langle 2^{4}\rangle = \langle 7\rangle$ as the subgroup of order 5
- $i = 10 : \langle 2^{2}\rangle = \langle 4\rangle$ as the subgroup of order 10
- $i = 20 : \langle 2^{1}\rangle = \langle 2\rangle$ as the subgroup of order 20

---

# Hard

---

12. As in 11 (parts $b$ and $c$), if we have a generator $g$ for the cyclic group $G = \langle g\rangle$ and $\text{ord}(G) = p_1^2 p_2$, we have

$$\phi(\text{ord}(G)) = \phi(p_1^2 p_2) = (p_1 - 1)p_1(p_2 - 1)$$

and we can find every subset $H$ of order $d$ such that $d|\phi(\text{ord}(G))$ as

$$\langle g^{\frac{\phi(\text{ord}(G))}{d}} \rangle = \text{ subgroup of } G \text{ of order } d$$

13. Assume first that $a \geq b$. As hinted, we note that if $a = qb + r$ with $q > 0$ and $0 \leq r < b$, then

$$a = qb + r > 1 \cdot r + r = 2r,$$

i.e. $r < a/2$. We note that in the calculation of $\gcd(a, b)$ using Euclids algorithm. i.e.

$$\begin{aligned} a_0 &= a \\ b_0 &= b \\ a_{n+1} &= b_n \\ b_{n+1} &= a_n \bmod b_n \end{aligned}$$

we have $a_{n+2} = b_{n+1} = a_n \bmod b_n$. Thus $a_{n+2} < a_n/2$ during the whole algorithm (except possibly at the first step, if we allow $a < b$; then $q$ would be 0 and $a$ and $b$ are exchanged).

Since we work with integers starting with $a \leq 2^n$ and $a$ halved every two steps, the algorithm cannot continue for more than $n$ double-steps, i.e. $2n$ steps.

In fact, we will stop before $n$ double-steps, since we have strict inequality in $a_{n+2} < a_n/2$. On the other hand, if $b > a$ we will get an extra initial step swapping $a$ and $b$.

14. The general idea here is that the adversary should use the extended Euclid's algorithm on $e_A$ and $e_B$ to get $(n, s, t)$ such that $se_a + te_b = n$, where $n$ is the gcd of $e_A$ and $e_B$. If the two ciphertexts are $c_A$ and $c_B$, the adversary computes $c_A^s \cdot c_B^t$. We have (in $\mathbb{Z}_N$)

$$c_A^s \cdot c_B^t = m^{se_A + te_b} = m^n.$$

In the common case (with small encryption exponents) that $e_A$ and $e_B$ are relatively prime (i.e., $n = 1$), the adversary has actually managed to find $m$.

15. Note first that even though $m = m_1 \cdot m_2$, the adversary does not know this factorization. However, we must have $c = m_1^e \cdot m_2^e$, which leads to the following attack.

    (a) Compute $(m_1^e, m_1)$ for all $2^{28}$ values of $m_1$ and store in a hash table with $m^e$ as key.

    (b) Compute $c/m_2^e$ for all $2^{28}$ values of $m_2$ and try to look it up in the table. When found, $m = m_1 \cdot m_2$.

    The time for each of these two phases is $2^{28}$ "steps" where a step is two encryptions, two hash table operations and one division. Storage requirement is $2^{28} \cdot (1024 + 28)$ bits, i.e. ca 32 Gbytes. Both requirements are clearly feasible and we see here another example of the insecurity of textbook RSA.

    **Remark:** What about using DES for session encryption? We note that

    - if we only need to keep data secret during the session, it is probably safe; breaking DES is feasible but still resquires substantial time and resources.
    - the adversary may record communication and break encryption later, so if the communication is important enough and must be confidential also in the future, DES is not a good choice.
    - DES is slower than AES, in addition to being insecure, so there seems to be no good reason for using it.

16. (a) Let $z = g^{(p-1)/2}$. Since $p$ is a large prime, it is odd and hence $(p-1)/2$ is an integer. Further, we have

$$z^2 = g^{p-1} = 1 \bmod p,$$

    by Fermat's little theorem. Thus $z^2 - 1 = (z+1)(z-1) = 0 \bmod p$, i.e. $(z+1)(z-1) = kp$ for some $k$. It follows that one of $z+1$ and $z-1$ must be a multiple of $p$. In the former case $z = p-1 \bmod p$, in the latter $z = 1 \bmod p$. But $z = 1 \bmod p$ is not possible, since that contradicts that $g$ is a generator for $\mathbb{Z}_p^*$.

    (b) We have that

$$y^{(p-1)/2} = (g^x)^{(p-1)/2} = (g^{(p-1)/2})^x = (p-1)^x.$$

    But $(p-1)^x$ assumes only the values 1 and $p-1$, depending on whether $x$ is even or odd. So, computing $y^{(p-1)/2}$ reveals the parity of $x$.

    **Remark:** This is a special case of the Pohlig-Hellman method, discussed in lecture 8.

## Think

17. The set `int` is just another choice of representatives for the elements of $\mathbb{Z}_{2^w}$. As an example, in $\mathbb{Z}_{16}$ we have $9 = 25 = 41 = -7 = -23 = \ldots$. In this course we have chosen to represent each $k$ in $\mathbb{Z}_{16}$ by a value between 0 and 15, inclusive. One could equally well have chosen a value between -8 and 7 (or any other range with 16 consecutive numbers, such as betwen 377 and 392).

    For the general case $w$, words in such a computer can represent natural numbers $k$ in the range $0 \leq k \leq 2^w - 1$. Now we think of a number $k$ in the range $2^n \leq k \leq 2^w - 1$ as the negative number $k - 2^w$, which is equal to $k$ in $\mathbb{Z}_{2^w}$.

Some examples, for the unrealistically small $w = 4$:

| $k$ in $\mathbb{Z}_{16}$, in binary | $k$ in $\mathbb{Z}_{16}$, in decimal | $k$ in `int` |
|---|---|---|
| 0101 | 5 | 5 |
| 1001 | 9 | 9-16=-7 |
| 1111 | 15 | 15-16 = -1 |

We can then make two observations:

(a) If the CPU is assumed to compute correctly in $\mathbb{Z}_{2^w}$, it follows automatically that it computes correctly in `int`, since we are just using other representatives for some numbers.

(b) A simple way to describe how a negative number $-k$ (where $k > 0$) in `int` is represented as a bit pattern is thus that it is the bit pattern of $-k + 2^w$. However, now we want to describe this representation without reference to $\mathbb{Z}_{2^w}$, as is often done in books on computer architecture (under the name of the two-complement representation).
For this, we rewrite $-k + 2^w$ in a somewhat complicated way, using $n = w - 1$.

$$-k + 2^w = 2^n + ((2^n - 1) - (k - 1)),$$

which easily checked using $2^w = 2^n + 2^n$. So now we can see that the bit pattern of $-k$ is

- a '1' as the most significant bit (because of $2^n + \ldots$),
- followed by the bitwise complement of $k - 1$ (because $(2^n - 1) - x$ as a bit pattern with $n$ bits is the bitwise complement of the bit pattern of $x$).

Finally, it follows that computations in `int` for $w = 32$ like

$$1234567890 \cdot 2 = -1825831516,$$

as you would get in many programming languages, are actually correct, when interpreted in $\mathbb{Z}_{2^{32}}$. If you choose our standard representative of the result, i.e. $-1825831516 + 2^{32} = 2469135780$ we see that the result is correct.

18. The reason why we use primes of the same size is connected to several attacks on poorly chosen primes, like Wiener's attack.
The other problem is that RSA's security is directly connected to **integer factorization hardness** and so every algorithm for integer factorization can be used as an attack on RSA and, some of these algorithms, can be more effective if, for example, *the two primes are of similar size*, like Fermat's factorization method.

The reasons that we don't want $p$ and $q$ to be **non-**prime numbers are (there can be even more reasons):

- To achieve the same security with respect to the case where $p, q$ are primes, we need to choose larger $p, q$. Otherwise, factoring is easier since there are *"many more factors that we can find"*[2]
- $\phi(N)$ is smaller and so there are less secret keys to randomly choose from
- *Connected to factorization:* it is more probable to pick a random message $m$ that is a divisor of $N$

But this doesn't mean that you are limited to using exactly two big primes!

[2] You have more divisors for $N$, and so it is easier to find them.