What we do in this lecture. In the previous lecture, we showed:

• how to encode a message, once we know the public code, i.e., the integers $n$ and $e$, and

• how to decode a message, if we know the secret code $d$.

In this lecture, we show how to generate the public code and the secret code.

How do we generate the public code: how to generate $n$. First, we generate two huge prime numbers $p$ and $q$. How to generate prime numbers is beyond the scope of this lecture. These numbers are kept secret.

The integer $n$ is then the product $p \cdot q$.

How to generate the second part $e$ of the public code: important reminders. The algorithm for generating $e$ uses the notions of:

• the least common multiple and

• the greatest common divisor.

The least common multiple: reminder. The least common multiple $\text{lcm}(a, b)$ of two numbers $a$ and $b$ is the smallest number that is divisible by both $a$ and $b$.

For example, if $a = 4$ and $b = 10$, then the smallest possible number that divides both 4 and 10 is 20:

• If we divide 20 by 10 or divide 20 by 4, we do not get any remainders.

• However, if we divide a number smaller than 20 by 10 or by 4, then at least in one of these divisions we will have a non-zero remainder.

The notion of the least common multiple is related to adding fractions:

• the usual way to add two fractions $\frac{n_a}{a} + \frac{n_b}{b}$ with denominators $a$ and $b$ is

• to bring them to the least common denominator – which is exactly $\text{lcm}(a, b)$.

The greatest common divisor: reminder. The greatest common divisor $\text{gcd}(a, b)$ of two numbers $a$ and $b$ is the largest number that divides both $a$ and $b$.

For example, if $a = 4$ and $b = 10$, then $\text{gcd}(4, 10) = 2$. Indeed:
• both 4 and 10 are divisible by 2, but
• if you have any larger number (e.g., 3 or 4), then either 4 or 10 will not be divisible by this larger number.

How to generate the second part \(e\) of the public code: idea. Based on \(p\) and \(q\), we generate the least common multiple \(\lambda = \text{lcm}(p - 1, q - 1)\) of the values \(p - 1\) and \(q - 1\). This value is computed as the ratio

\[
\text{lcm}(p - 1, q - 1) = \frac{(p - 1) \cdot (q - 1)}{\gcd(p - 1, q - 1)},
\]

where the greatest common divisor \(\gcd(p - 1, q - 1)\) is computed by using Euclid’s algorithm. This algorithm is presented later in this text.

Then, we pick up an integer \(e < \lambda\) for which \(\gcd(e, \lambda) = 1\), i.e., that has no common divisors with \(\lambda\). Usually, this number is larger than 3 and smaller than \(10^5\). It can be selected, e.g., by trying all the values starting with \(e = 4\).

How to generate the second part \(e\) of the public code: details. The value \(e\) is used to encode the messages. As we have seen in the previous lecture, the larger the value \(e\), the more binary digits this number has, the longer it take to compute the encoding

\[c = m^e \mod n.\]

Similarly, the larger the value \(d\), the longer it takes to decode. So which value \(e\) should we use?

Let us consider a typical situation that requires computer security: buying something from amazon.com or from any other online company. The online company has many powerful and fast computers – otherwise, it would not be able to handle many purchasing requests that millions of customers send. These powerful computers make it easy for the company to decode all the data. On the other hand, the company is interested in serving all potential customers, including customers who do not have easy access to powerful computers. If the number \(e\) used for encoding is too large, this may cause an unnecessary delay in the customer communication with the company – and, as a result, the customer will switch to a competitor. In such situations, it is desirable to make the value \(e\) as small as possible.

The smallest possible integer \(e\) is 1, but raising \(m\) to the power 1 does not encode anything. The next smallest value is \(e = 2\), but for the algorithm to work, we need \(\gcd(e, \lambda) = 1\), and here, since both \(p - 1\) and \(q - 1\) are even number, their \(\text{lcm}\ \lambda\) is also even. So, both \(e\) and \(\lambda\) have a common divisor 2.

The next smallest value is \(e = 3\); however, it is not recommended since for \(e = 3\), it is sometimes possible to easily decode the encoded messages. So, the usual recommendation is to use the smallest \(e > 3\) for which \(\gcd(e, \lambda) = 1\).

Comment. This is the recommendation that we will follow in this lecture.

However, it is necessary to mention that in some applications, it is necessary to minimize not the encoding time, but the decoding time. For example, if we
are controlling a drone with limited energy and limited computational ability, then it is OK to spend time on encoding, but decoding should be as fast as possible. In this case, we select $d$ to be the smallest integer $d > 3$ for which $\gcd(d, \lambda) = 1$, and find $e$ based on $d$ the same way in the usual case, we find $d$ based on $e$.

**How to compute $\gcd(a, b)$: Euclid’s algorithm – idea.** Without losing generality, we can assume that $a < b$. Then, if we divide $b$ by $a$, we get $b = k \cdot a + r$ for some remainder $r$.

- Clearly, if some number divides both $a$ and $b$, then it also divides the remainder $r = b - k \cdot a$.
- Vice versa, if some number divides $r$ and $a$, then it also divides $b = k \cdot a + r$.

Thus, common divisors of $a$ and $b$ are exactly the same as common divisors of $r$ and $a$. This means that the greatest common divisor $\gcd(a, b)$ of $a$ and $b$ is equal to the greatest common divisor $\gcd(r, a)$ of $r$ and $a$: $\gcd(a, b) = \gcd(r, a)$.

This leads to the following algorithm.

**Euclid’s algorithm.** We start with $a < b$. Then, we divide $b$ by $a$.

- If there is no remainder (i.e., if the remainder is 0), this means that $a$ is the greatest common divisor of $a$ and $b$: $\gcd(a, b) = a$.
- If there is a non-zero remainder $r$, then we apply the same algorithm to compute $\gcd(r, a)$ and take $\gcd(a, b) = \gcd(r, a)$.

At each stage, the numbers decrease, so this process will stop – and generate the desired $\gcd(a, b)$.

**Example.** Let us compute $\gcd(105, 2020)$.

- First, we divide $2020$ by $105$, we get $$2020 = 19 \cdot 105 + 25 \ (= 1995 + 25).$$
The remainder is $r = 25$, so we need to compute the value $\gcd(25, 105)$.
- Now, to compute $\gcd(25, 105)$, we divide $105$ by $25$ and get $$105 = 4 \cdot 25 + 5.$$ Here, the remainder is $r = 5$, so we need to compute $\gcd(5, 25)$.
- Finally, we divide $25$ by $5$, we get $$25 = 5 \cdot 5 + 0.$$ There is no remainder, so $\gcd(5, 25) = 5.$
This value 5 is the desired answer:

\[
\gcd(105, 2020) = 5.
\]

**Practice.** Practice applying this algorithm to other integers.

**How to generate the secret code** \( d \). We want to find a number \( d \) for which

\[
d \cdot e = 1 \mod \lambda.
\]

By definition of the remainder, this means that we need to find a number \( d \) for which

\[
d \cdot e = k \cdot \lambda + 1
\]

for some integer \( k \). This equality is equivalent to

\[
1 = d \cdot e - k \cdot \lambda.
\]

In other words, what we need is to represent 1 as a linear combination of numbers \( e \) and \( \lambda \) with integer coefficients. To make it clearer, we can emphasize the two numbers \( e \) and \( \lambda \) by underlining them:

\[
1 = d \cdot e - k \cdot \lambda.
\]

The coefficient at \( e \) in this representation will be the desired value \( d \).

Such a representation of 1 as a linear combination of \( e \) and \( \lambda \) can be obtained by the following modification of Euclid’s algorithm.

Let us illustrate this modification on the example of two numbers that do not have any common divisors: \( \lambda = 37 \) and \( e = 7 \). Let us first show how Euclid’s algorithm work on these two numbers.

• First, we divide 37 by 7, and get

\[
37 = 5 \cdot 7 + 2. \tag{1}
\]

• The remainder is 2, so we divide 7 by 2:

\[
7 = 3 \cdot 2 + 1. \tag{2}
\]

• Now, we are left with 2 and 1. Of course, 1 divides 2, so 1 is the greatest common divisor of 1 and 2.

Thus, the greatest common divisor of 37 and 7 is also 1, i.e., 37 and 7 do not have a common divisor.

Let us show how, in this case, we can generate the desired value \( d \).

• From the equation (1), by moving the term proportional to 7 to the other side, we conclude that the remainder 2 can be represented as a linear combination of the original numbers 7 and 37. In the resulting formula, we underline these two numbers to emphasize that they are the two original numbers:

\[
2 = 1 \cdot 37 - 5 \cdot 7. \tag{1a}
\]
From the second equation (2), we can similarly represent 1 as a linear combination of 7 and 2:

\[ 1 = 1 \cdot 7 - 3 \cdot 2. \]  (2a)

We know, from the formula (1a), that 2 can also be represented as a linear combination of 7 and 37. If we substitute the expression (1a) for 2 into the formula (2a), we get the representation of the value 1 as a linear combination of the original two values 37 and 7:

\[ 1 = 1 \cdot 7 - 3 \cdot (1 \cdot 37 - 5 \cdot 7) = 1 \cdot 7 - 3 \cdot 37 + 15 \cdot 7. \]

Now, we combine the terms proportional to 7 (there is only term proportional to 37, no need to combine it) and get

\[ 1 = 16 \cdot 7 - 3 \cdot 37. \]

If we move the term proportional to 37 to the other side, we conclude that

\[ 16 \cdot 7 = 3 \cdot 37 + 1. \]

Thus, if we divide 16 · 7 by 37, the remainder is 1. So, we got the value \( D = 16 \) for which \( D \cdot e = 1 \mod 37 \).

As \( d \), we take the remainder of dividing \( D \) by \( \lambda \). In this case, \( 16 = 0 \cdot 37 + 16 \), so the remainder is \( d = 16 \).

**Comment.** In general, after we get a value \( D \) for which \( D \cdot e = 1 \mod \lambda \), we take as \( d \) the remainder of dividing \( D \) by \( \lambda \). For example, if \( \lambda = 37 \), then:

- for \( D = 80 \), since \( 80 = 2 \cdot 37 + 6 \), we take \( d = 6 \);
- for \( D = -15 \), since \( -15 = (-1) \cdot 37 + 22 \), we take \( d = 22 \).

**Practice.**

- Practice this algorithm on some other example.
- Now that we know how to generate the public code \((n, e)\) and the secret code \(d\) and how to encode and decode messages, practice the whole process.

**Example of the whole process.** Let us take \( p = 5 \) and \( q = 7 \).

**Example of the whole process: computing \( n \).** Then \( n = p \cdot q = 35 \).

**Example of the whole process: computing \( \lambda \).** Let us apply Euclid’s algorithm to compute \( \gcd(p - 1, q - 1) = \gcd(4, 6) \).

- First, we divide 6 by 4 and get 6 = 1 · 4 + 2, with remainder 2.
- Thus, we need to compute \( \gcd(4, 2) \). For this, we divide 4 by 2. There is no remainder, so \( \gcd(4, 6) = \gcd(2, 4) = 2 \).
Thus, \[ \lambda = \text{lcm}(4, 6) = \frac{4 \cdot 6}{\text{gcd}(4, 6)} = \frac{24}{2} = 12. \]

**Example of the whole process:** computing \( e \). Let us select the smallest value \( e > 3 \) for which \( \text{gcd}(e, 12) = 1 \).

**First try:** \( e = 4 \). We start with \( e = 4 \). For \( e = 4 \), to compute \( \text{gcd}(4, 12) \), we divide 12 by 4. There is no remainder, so \( \text{gcd}(4, 12) = 4 \), and we want 1.

**Next try:** \( e = 5 \). So, we try the next value \( e = 5 \).

- For this value, to compute \( \text{gcd}(5, 12) \), we divide 12 by 5 and get
  \[ 12 = 2 \cdot 5 + 2, \]  
  with remainder 2.

- Thus, we need to compute \( \text{gcd}(2, 5) \). To compute this value, we divide 5 by 2 and get
  \[ 5 = 2 \cdot 2 + 1. \]  
  The remainder is 1, so \( \text{gcd}(5, 12) = \text{gcd}(2, 5) = 1. \)

**Example of the whole process:** resulting public key. So, we have generated the public key: \( n = 35 \) and \( e = 5 \).

**Example of the whole process:** computing \( d \). Let us now generate the secret key \( d \).

- From the equation (3), we conclude that
  \[ 2 = 1 \cdot 12 - 2 \cdot 5. \]  
  (3a)

- From the equation (4), we get
  \[ 1 = 1 \cdot 5 - 2 \cdot 2. \]  
  (4a)

Substituting the expression (3a) for 2 into the formula (4a), we conclude that
\[ 1 = 1 \cdot 5 - 2 \cdot (1 \cdot 12 - 2 \cdot 5) = 1 \cdot 5 - 2 \cdot 12 + 4 \cdot 5 = 5 \cdot 5 - 2 \cdot 12. \]

Thus, \( 5 \cdot 5 = 2 \cdot 12 + 1 \).

So, for \( d = 5 \), we have \( d \cdot e = 1 \mod 12 \). Thus, we have also generated a secret key \( d = 5. \)
Comment. It so happens that here \( d = e \), since this is a toy example. In general, \( d \) and \( e \) are different.

**Example of the whole process: transforming \( e \) and \( d \) into binary.** For the purposes of encoding and decoding, we need to know the binary extension of \( e \) and \( d \). This can be obtained by using the general algorithm:

\[
\begin{align*}
5 &= 2 \cdot 2 + 1; \\
2 &= 1 \cdot 2 + 0; \\
1 &= 0 \cdot 2 + 1. \\
\end{align*}
\]

So, \( e = d = 101 \).

**Example of the whole process: encoding a message.** Let us take any message, e.g., \( m = 3 \). Then, to encode it, we compute \( m_0 = m = 3 \), then

\[
\begin{align*}
m_1 &= 3 \cdot 3 \mod 35 = 9 \mod 35 = 9; \\
m_2 &= 9 \cdot 9 \mod 35 = 81 \mod 35 = 11;
\end{align*}
\]

(here, \( 81 = 2 \cdot 35 + 11 = 70 + 11 \)).

Here, \( e = 4 + 1 = 2^2 + 2^0 \), so

\[
m^e = m^4 \cdot m^1 = m^2 \cdot m^0 = m_2 \cdot m_0.
\]

In this case,

\[
c = m_2 \cdot m_0 \mod 35 = 11 \cdot 3 \mod 35 = 33 \mod 35 = 33.
\]

Thus, the encoded message is

\[
c = 33.
\]

**Example of the whole process: decoding a message.** Let us show his this message will be decoded. Here, \( m_0 = c = 33 \),

\[
m_1 = 33 \cdot 33 \mod 35 = 1089 \mod 35 = 4;
\]

(since \( 1089 = 31 \cdot 35 + 4 = 1085 + 4 \)),

\[
m_2 = 4 \cdot 4 \mod 35 = 16 \mod 35 = 16.
\]

Here, \( d = 4 + 1 = 2^2 + 2^0 \), so

\[
c^d = c^4 \cdot c^1 = c^2 \cdot c^0 = m_2 \cdot m_0.
\]

In this case,

\[
m = m_2 \cdot m_0 \mod 35 = 16 \cdot 33 \mod 35 = 528 \mod 35 = 3.
\]

(since \( 528 = 15 \cdot 35 + 3 = 525 + 3 \)).

Thus, we indeed get back the original message \( m = 3 \).