Homework 05

HOMEWORK 05, DUE BY 2022-03-03

Question 1: Define the Fibonacci sequence as follows:

$$\begin{cases} F_0 = 0, \\ F_1 = 1, \\ F_{n+2} = F_{n+1} + F_n, \text{ where } n \geq 0 \end{cases}$$

Prove the following statement for any $n \in \mathbb{Z}^+$:

$$\sum_{k=1}^{n} F_{2k} = F_{2n+1} - 1.$$

Answer:

Write the first few terms of this sequence: $F_0 = 0,\ F_1 = 1,\ F_2 = 1,\ F_3 = 2,\ F_4 = 3, \ldots$.

Induction Base: $n = 1$. In this case $\sum_{k=1}^{1} F_{2k} = F_2 = 1$. It is indeed equal $F_{2n+1} - 1 = 2 - 1 = 1$.

Inductive Step: $n = m \ \Rightarrow \ n = m + 1$.

Inductive hypothesis:

$$\sum_{k=1}^{m} F_{2k} = F_{2m+1} - 1.$$

We should verify same about the next sum:

$$\sum_{k=1}^{m+1} F_{2k} = \left(\sum_{k=1}^{m} F_{2k} \right) + F_{2m+2} = (F_{2m+1} - 1) + F_{2m+2} = \\ = (F_{2m+1} + F_{2m+2}) - 1 = F_{2m+3} - 1.$$

The inductive step is now proven. Here we applied the inductive hypothesis and also the formula for $F_{2m+3} = F_{2m+2} + F_{2m+1}$ (the definition of Fibonacci numbers).

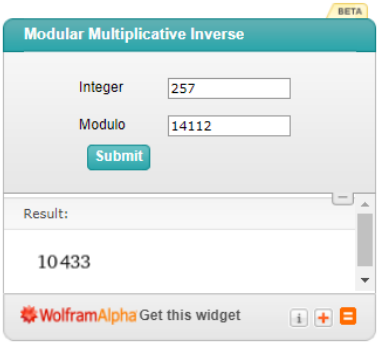
Question 2: What is the original message encrypted using the RSA cryptosystem with $N = 113 \cdot 127$, and public exponent $e = 257$, if the encrypted message is `[1024, 12523, 10691]`. Each number in the list is to be decrypted separately by raising to the power of *private exponent* d modulo N.

To decrypt, first find the decryption exponent d which is the multiplicative inverse of e modulo $\varphi(N)$.

Note: Multiplicative inverses can be computed using the widget https://bit.ly/3rYFELL or a Python function `mod_inverse(...)` (see source code in Question 6 below). RSA cryptography is described in (Rosen2019), page 316.

Answer:

Find the private exponent d that is the multiplicative inverse of the public exponent $e = 257$ modulo $\varphi(N) = \varphi(113 \cdot 127) = (113-1)(127-1) = 14112$. According to the Wolfram Alpha: $257^{-1} = 10433$. We can verify: $257 \cdot 10433 \equiv 1 \pmod{14112}$.



To decrypt the message `[1024, 12523, 10691]`, raise every term in this list to power $d = 10433$ modulo $N = 113 \cdot 127 = 14351$.

```
>>> enc_message = [1024, 12523, 10691]
>>> [m**10433 % 14351 for m in enc_message]
[4, 5, 1990]
```

So the decrypted message is the list of numbers `[4, 5, 1990]`.

Note: Using this public key $(N, d) = (14351, 257)$ one could encrypt arbitrary sequences of integers in the interval $[1; 14350]$, including long text messages, archive files or images encoded using these numbers. In practice running RSA algorithm (or a similar public key cryptosystem) is less efficient than a symmetric key algorithm (such as AES) – RSA is too slow to for large amount (many megabytes) of data.

For this reason RSA is typically used to exchange the symmetric keys for further communication, certificates, digital signatures and other small-size items. Also the primes used in RSA cryptography should be much larger than in this example to make the brute force factorization of N impossible.

Question 3: This problem could be tedious to compute on paper; consider using Python scripts or something similar.

(A) For each number $a \in \{1, 2, 3, \ldots, 18\}$ find its *multiplicative order* modulo 19: it is the smallest positive integer k such that $a^k \equiv 1 \pmod{19}$. Fill in all entries in this table:

a	1	2	3	4	$\cdots$	17	18
Multiplicative order of a	1				$\cdots$		

(B) For every number $a \in a \in \{1, 2, 3, \ldots, 18\}$ which is **not** a primitive root (modulo 19) find the discrete logarithm (index) of a to the base $r = 2$ modulo 19: $\operatorname{ind}_2 a \pmod{19}$.

(C) Alice and Bob have selected prime number $p = 19$ and a primitive root $r = 2$ for their Diffie-Hellman key exchange. Alices secret exponent is $a = 11$, but Bobs secret is $b = 13$. Compute the message $A \equiv r^a \pmod{p}$ that Alice sends to Bob, and also the message $B \equiv r^b \pmod{p}$ that Bob sends to Alice.

Compute the secret key they have both agreed to in this key exchange. Diffie-Hellman key exchange is defined in (Rosen2019), page 319.

Answer:

(A) Fill in the table with multiplicative orders:

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Multiplicative order of a	1	18	18	9	9	9	3	6	9	18	3	6	18	18	18	9	9	2

```
>>> [2**k % 19 for k in range(1,19)]
[2, 4, 8, 16, 13, 7, 14, 9, 18, 17, 15, 11, 3, 6, 12, 5, 10, 1]
>>> [3**k % 19 for k in range(1,19)]
[3, 9, 8, 5, 15, 7, 2, 6, 18, 16, 10, 11, 14, 4, 12, 17, 13, 1]
>>> [4**k % 19 for k in range(1,19)]
[4, 16, 7, 9, 17, 11, 6, 5, 1, 4, 16, 7, 9, 17, 11, 6, 5, 1]
>>> [5**x % 19 for x in range(1,19)]
[5, 6, 11, 17, 9, 7, 16, 4, 1, 5, 6, 11, 17, 9, 7, 16, 4, 1]
>>> [6**x % 19 for x in range(1,19)]
[6, 17, 7, 4, 5, 11, 9, 16, 1, 6, 17, 7, 4, 5, 11, 9, 16, 1]
>>> [7**x % 19 for x in range(1,19)]
[7, 11, 1, 7, 11, 1, 7, 11, 1, 7, 11, 1, 7, 11, 1, 7, 11, 1]
```

To find the multiplicative orders, we can compute the powers of numbers 2^k, 3^k, 4^k (modulo 19). Wait until we find the first number 1 in each sequence.

(B) Display all the powers of 2:

$$2, 4, 8, 16, 13, 7, 14, 9, 18, 17, 15, 11, 3, 6, 12, 5, 10, 1.$$

Find all the primitive roots in the table found earlier (they have multiplicative order equal to 18):

$$\begin{cases} \operatorname{ind}_2 2 = 1 \pmod{19} \\ \operatorname{ind}_2 3 = 13 \pmod{19} \\ \operatorname{ind}_2 10 = 17 \pmod{19} \\ \operatorname{ind}_2 13 = 5 \pmod{19} \\ \operatorname{ind}_2 14 = 7 \pmod{19} \\ \operatorname{ind}_2 15 = 11 \pmod{19} \end{cases}$$

Note: Observe that all the discrete logarithm values for the primitive roots are $\{1, 13, 17, 5, 7, 11\}$ – namely, they are numbers x such that $\gcd(x, 18) = 1$ (numbers not divisible by 2 or by 3). Thus the number of primitive roots for $p = 19$ equals :math;`varphi(18)`.

This is true also in the general case: For every prime p, the number of primitive roots is $\varphi(p-1)$.

(C) Let $a = 11$ and $b = 13$ be the secret keys selected by Alice and Bob respectively.

- Alice sends to Bob the number $A = 2^{11} \bmod 19 = 15$.
- Bob sends to Alice the number $B = 2^{13} \bmod 19 = 3$.

After receiving the numbers A and B respectively:

- Alice computes $B^a \bmod p = 3^{11} \bmod 19 = 10$.
- Bob computes $A^b \bmod p = 15^{13} \bmod 19 = 10$.

Therefore number 10 is the shared secret key (it is computed by both Bob and Alice).

Note: If prime number is small (such as $p = 19$), this key exchange is not secure, since it is possible to compute discrete logarithms such as $\operatorname{ind}_r A = a \bmod p$ and $\operatorname{ind}_r B = b \bmod p$. It is doable even by raising r to all possible powers.

For example, by looking at the number sequence in (B) one can find that $\text{ind}_2 15 = 11 \bmod 19$ (11 is Alices secret number). But if p is very large number (tens or hundreds of digits), this is not effectively doable.

Question 4: Prove by mathematical induction that the number $k^{2^n} - 1$ is divisible by 2^{n+2}, if k is a positive odd integer, but n is any positive integer.

Answer:

Fix any positive odd integer k, but prove the statement doing induction by n.

Inductive Base: $n = 1$. We check that $k^{2^1} - 1 = k^2 - 1 = (k-1)(k+1)$. Let k be an odd integer.

Both numbers $k-1$ and $k+1$ are even. At least one of them is also divisible by 4 (depending on whether $k \equiv 1 \pmod 4$ or $k \equiv 3 \pmod 4$). If we multiply an even number (say, $k-1$) by a number divisible by 4 (say, $k+1$), their product is divisible by $2 \cdot 2^2 = 2^3 = 8$.

So, the base is true: $k^{2^1} - 1$ is divisible by $2^{1+2} = 2^3 = 8$.

Inductive Step: $n = m \Rightarrow n = m+1$.

Inductive hypothesis: $k^{2^m} - 1$ is divisible by 2^{m+2}.

We should verify that $k^{2^{m+1}} - 1$ is divisible by 2^{m+3}. Let us do algebraic transformation:

$$k^{2^{m+1}} - 1 = k^{2 \cdot 2^m} - 1 = \left(k^{2^m}\right)^2 - 1^2 = \left(k^{2^m} - 1\right)\left(k^{2^m} + 1\right).$$

The first parenthesis in this product is divisible by 2^{m+2} by inductive hypothesis. The second parenthesis $\left(k^{2^m} + 1\right)$ is an even number (as the sum of two odd numbers).

Therefore their product is divisible by $2^{m+2} \cdot 2 = 2^{m+3}$. This concludes the inductive step.

Question 5: Consider the following two propositions:

Proposition1: Let $n \in \mathbb{Z}^+$ be any positive integer. Then the following inequality is true:

$$\prod_{k=1}^{n} \frac{2k-1}{2k} \leq \frac{1}{\sqrt{3n}}.$$

Proposition2: Let $n \in \mathbb{Z}^+$ be any positive integer. Then the following inequality is true:

$$\prod_{k=1}^{n} \frac{2k-1}{2k} \leq \frac{1}{\sqrt{3n+1}}.$$

(A) Choose any of the two propositions and prove it using mathematical induction.

(B) Is any of the two propositions false, or are they both true? Justify your answer.

Answer:

(A) Let us prove the second inequality involving $\frac{1}{\sqrt{3n+1}}$ (rather than $\frac{1}{\sqrt{3n}}$).

Inductive Base: $n = 1$. Verify that $\frac{1}{2} \leq \frac{1}{3 \cdot 1 + 1} = \frac{1}{2}$. So it is valid.

Inductive Step: $n = m \Rightarrow n = m+1$. We write the inductive hypothesis – assume that this is true:

$$\prod_{k=1}^{m} \frac{2k-1}{2k} \leq \frac{1}{\sqrt{3m+1}}.$$

Let us prove another inequality:

$$\prod_{k=1}^{m+1} \frac{2k-1}{2k} \leq \frac{1}{\sqrt{3(m+1)+1}}.$$

Please note that the latter is the inequality we need to prove (cannot assume that it is true). We proceed by rewriting this (unproven) inequality into a sequence of inequalities. We will prove the last one and then will proceed backwards to prove the initial inequality as well.

$$\prod_{k=1}^{m+1} \frac{2k-1}{2k} \leq \frac{1}{\sqrt{3(m+1)+1}},$$
$$\frac{1}{2} \cdot \frac{3}{4} \cdot \ldots \cdot \frac{2m-1}{2m} \cdot \frac{2m+1}{2m+1} \leq \frac{1}{\sqrt{3m+4}},$$
$$\left(\prod_{k=1}^{m} \frac{2k-1}{2k}\right) \cdot \frac{2m+1}{2m+1} \leq \frac{1}{\sqrt{3m+4}},$$
$$\frac{1}{\sqrt{3m+1}} \cdot \frac{2m+1}{2m+1} \leq \frac{1}{\sqrt{3m+4}},$$
$$\frac{2m+1}{2m+2} \leq \frac{\sqrt{3m+1}}{\sqrt{3m+4}},$$
$$\frac{4m^2+4m+1}{4m^2+8m+4} \leq \frac{3m+1}{3m+4},$$
$$(3m+4)(4m^2+4m+1) \leq (3m+1)(4m^2+8m+4),$$
$$12m^3+12m^2+3m+16m^2+16m+4 \leq 12m^3+24m^2+12m+4m^2+8m+4,$$
$$12m^3+28m^2+19m+4 \leq 12m^3+28m^2+20m+4,$$
$$0 \leq m.$$

The last inequality is true (and one can check that every next inequality also implies the one above it). So the inductive step is complete.

Note: If we try to prove the first inequality, then the inductive step would lead us to proving inequality which looks like this:

$$\frac{2m+1}{2m+2} \leq \frac{\sqrt{3m}}{\sqrt{3m+3}}.$$

This inequality is false for many values of m. For example, if $m = 1$, then $\frac{2\cdot1+1}{2\cdot1+2} > \frac{\sqrt{3\cdot1}}{\sqrt{3\cdot1+3}}$ or $\frac{3}{4} > \frac{\sqrt{3}}{\sqrt{6}} \approx 0.7071068$.

The fact that we could not complete the inductive step does not imply that the inequality $\prod_{k=1}^{n} \frac{2k-1}{2k} \leq \frac{1}{\sqrt{3n}}$ is false. In fact, this inequality is true (see (B)).

(B) Both inequalities are true.

If some expression is less or equal than $\frac{1}{\sqrt{3n+1}}$ then it must be less or equal than $\frac{1}{\sqrt{3n}}$ (the second number is larger). Formally:

$$\prod_{k=1}^{n} \frac{2k-1}{2k} \leq \frac{1}{\sqrt{3n+1}} \leq \frac{1}{\sqrt{3n}}.$$

Question 6: Assume that the Python code given below implements RSA encryption. The public key is (N, e), where $N = 125060292466904545847084304952630886510097084 7 = p \cdot q$ is the product of two unknown primes, but public exponent is $e = 257$.

Unfortunately, an attacker has gained access to the `decrypt(message)` function as well (see Python source). The decrypt function does not contain the private exponent, but it uses a different exponent d_1 that can reverse the encryption of the public exponent $e = 257$.

(A) Factorize the modulus number N – express it as a product of two primes.

(B) Recover the proper private exponent d such that $d \cdot e \equiv 1 \pmod{\varphi(N)}$ and also $d < N$.

Note: You may need to apply some knowledge how to attack RSA algorithm. See https://stanford.io/3I6fQTG. RSA is still considered safe, if used properly; it is a popular public key method for encryption and electronic signatures. The above article summarizes known attacks discovered during 20 years of research.

For example, consider the following result (Page 3 of the article):

Fact 1: Let $(N; e)$ be an RSA public key. Given the private key d, one can efficiently factor the modulus $N = p \cdot q$. Conversely, given the factorization of N, one can efficiently recover d.

```
N = 125060292466904545847084304952630886510097084 7
e = 257

# A utility method to find multiplicative inverse a^(-1) (modulo n)
def mod_inverse(a, n):
    a1, v1 = 0, n
    a2, v2 = 1, a
    while v2 != 0:
        k = v1 // v2
        a1, a2 = a2, a1 - a2 * k
        v1, v2 = v2, v1 - v2 * k
    if a1 < 0:
        a1 += n
    return a1

# Public encryption function -- should be safe to show to everyone.
def encrypt(message):
    return pow(message, e, N)

# decrypt() is the inverse of encrypt(); exposes the RSA private key to attackers.
# (Even though the prime factors of N or the private exponent d<N are not shown.)
def decrypt(message):
    big_num = 1157020808057230654431538840923638721379001278990144 0
    d1 = mod_inverse(e, big_num)
    return pow(message, d1, N)

message = int(input("Enter a positive number to encrypt (up to 40 digits): "))
enc_message = encrypt(message)
print('Encrypted message: {}'.format(enc_message))
dec_message = decrypt(enc_message)
print('Decrypted message: {}'.format(dec_message))
```

This program lets the user enter a number (`message` variable), encrypts the number, then decrypts (should get back the original number).

```
Enter a number to encrypt (up to 40 digits): 2022
Encrypted message: 476636266497450731167878798199081427180792230
Decrypted message: 2022
```

Answer:

(A) First, output the exponent d_1 that was used for decryption. Modify the function `decrypt(message)` with a print statement:

```
def decrypt(message):
    big_num = 1157020808057230654431538840923638721379001278990144O
    d1 = mod_inverse(e, big_num)
    print('d1={}'.format(d1))
    return pow(message, d1, N)
```

This prints value $d_1 = 5672553378023776749353069803750135365515726114893313$.

Next, we proceed as follows (see *Fact 1* in the above article by Dan Boneh).

```
>>> N = 125060292466904545847084304952630886510097O847
>>> e = 257
>>> d1 = 5672553378023776749353069803750135365515726114893313
>>> # Find some multiple of phi(N)
>>> k = d1*e - 1
>>> k
1457846218152110624583738939563784788937541611527581440
>>> # Express "k" as (2**t)*r, where r is odd.
>>> k % 32
0
>>> k % 64
0
>>> k % 256
0
>>> k % 512
256
>>> We now know that k = (2**8)*r.
>>> r = k // 256
>>> r
5694711789656682127280230232671034331787271920029615
>>> # Generate random number g, where 1<g<N.
>>> import random
>>> g = random.randrange(1,N)
>>> g
7651207992071377080655030753769189779195101l4
>>> # Raise g**r (mod N).
>>> aa = pow(g,r,N)
>>> aa
7714083422661748061756589525832440049186049ll
>>> pow (aa,2,N)
1227309301600185490641898838018095109256204693
>>> pow (aa,4,N)
1131008743429714699663955356579232369519670028
>>> pow (aa,8,N)
1
>>> # since ((g**r)**8) == 1 (mod N), we have (g**r)**4 is a square root of 1
>>> import math
>>> # Suspect that (g**r)**4 == 1 (mod p), where "p" is one of the factors of "N".
>>> math.gcd(pow(aa,4,N)-1,N)
```

(continues on next page)

(continued from previous page)

```
11111111111111111111
>>> p = math.gcd(pow(aa,4,N)-1,N)
>>> N % p
0
>>> q = N//p
>>> q
112554263220214091262377
>>> N == p*q
True
```

We have successfully factorized the number N:

$$N = 11111111111111111111 \cdot 112554263220214091262377$$

Formally, let g be a random number from $(1; N-1)$ and let $r = (d_1 \cdot e - 1)/256 = 569471178965668212728023023267103433178727192002961 5$. (Here 256 is the largest power of two that divides $d_1 \cdot e - 1$.)

After that raise the randomly selected g to the power r and build this sequence:

$$a_0 = g^r \bmod N, \quad a_1 = a_0^2 \bmod N, \quad a_2 = a_1^2 \bmod N, \quad a_3 = a_2^2 \bmod N, \ldots$$

Every next term in this sequence is the square of the previous one (modulo N). This sequence is computed until its terms are all equal 1. Let i be the smallest index for which $a_i = 1$. Look at a_{i-1} (the last term of the sequence which differs from 1). If it is $-1 \equiv N-1$ then the attack was unsuccessful (and we pick another random number g and start anew).

If a_{i-1} has some value different from ± 1 then compute $p = \gcd(a_{i-1} - 1, N)$. (According to https://stanford.io/3I6fQTG, Page 3 this is one of the factors of N.)

(B) Once number N is factorized, it is easy to compute the standardized decryption key d.

Assume that the function `mod_inverse(a, n)` is defined (see the Python source text above).

```
>>> euler_phi = (p-1)*(q - 1)
>>> euler_phi
125060292466904545847071938415197753989859736 0
>>> N
125060292466904545847084304952630886510097084 7
>>> def mod_inverse(a, n):
...     a1, v1 = 0, n
...     a2, v2 = 1, a
...     while v2 != 0:
...         k = v1 // v2
...         a1, a2 = a2, a1 - a2 * k
...         v1, v2 = v2, v1 - v2 * k
...     if a1 < 0:
...         a1 += n
...     return a1
...
>>>
>>> d = mod_inverse(e, euler_phi)
>>> d
109975198823036682340226685143325651368514787 3
>>> d < N
True
```

We have $d = 109975198823036682340226685143325651368514787 3$. This shows that the cryptographic key (N, e) is broken – private key (exponent) d is found.