Project II
Κρυπτογραφία Δημοσίου Κλειδιού

Υφαντίδης Δημήτριος (ΑΕΜ: 3938)

2 Ιουνίου 2023

Εισαγωγή

Το παρόν αποτελεί την αναφορά του δεύτερου μέρους της εργασίας στο μάθημα "Θεμελιώσεις Κρυπτογραφίας".

Η αναφορά είναι γραμμένη σε LATEX και μεταγλωττίστηκε από τον **MiKTeX** compiler (ver: One MiKTeX Utility 1.7 - MiKTeX 23.5).

Οι υλοποιήσεις των ασκήσεων έγιναν σε γλώσσα **Python (v 3.10.6)**. Για κάθε άσκηση που ζητάει υλοποίηση σε κώδικα, υπάρχει ο αντίστοιχος φάκελος:

proj_2_crypto_3938/κώδικας/ex*

Όπου βρίσκεται ένα μοναδικό python script, **ex*.py** που περιέχει την υλοποίηση και ενδεχομένως κάποια ".txt" αρχεία ή άλλους πόρους για το πρόγραμμα.

Σχεδόν όλοι οι μαθηματικοί συμβολισμοί που εμφανίζονται στο κείμενο είναι κοινώς αποδεκτοί. Εξαίρεση αποτελούν οι ακόλουθοι, που διευκρινίζονται:

- $\mathbf{x} \bmod \mathbf{y}$: Το υπόλοιπο της ακέραιας διαίρεσης του x με το y, ενώ το $\mathbf{x} \pmod{\mathbf{y}}$ αποδίδεται σε κλάση ισοδυναμίας.
- $\mathbb{N}$: Το σύνολο των φυσικών αριθμών **συμπεριλαμβάνοντας** το 0, δηλ. [0, 1, 2, ...].
- $\mathbb{N}^*$: Το σύνολο των φυσικών αριθμών **χωρίς** το 0, δηλ. [1, 2, 3, ...].
- $\mathbb{P}$: Το σύνολο των πρώτων αριθμών.

Άσκηση 1 (6.1)

Υλοποιήθηκε ο αλγόριθμος (7.2.2) της υποενότητας (7.2.3) στη συνάρτηση `fast_pow_mod_m(b: int, e: int, m: int)`. Το πρωτόκολλο Diffie-Hellman απαιτεί τον υπολογισμό του κοινού κλειδού, τον ακέραιο $g^{ab}\, mod\, m$. Άρα, $b\ :=\ g,\ e\ :=\ a\cdot b,\ m\ :=\ m$ οι αναθέσεις στις παραμέτρους της προαναφερθούσας συνάρτησης.

Για τις δοσμένες τιμές, $(g,\ p,\ a,\ b) = (3,\ 101,\ 77,\ 91)$, παίρνουμε ως αποτέλεσμα:

$$k\ =\ 3^{7007}\, mod\, 101\ =\ 66$$

(Υλοποίηση: **κώδικας/ex1/ex1.py**)

Άσκηση 2 (7.3)

Η υλοποίηση είναι ίδια με αυτή της προηγούμενης άσκησης. Παίρνουμε ως αποτέλεσμα:

$$5^{77}\, mod\, 19\ =\ 9$$

(Υλοποίηση: **κώδικας/ex2/ex2.py**)

Άσκηση 3 (8.19)

Ζητείται ν.δ.ο:

$$p_n < 2^{2^n},\quad \forall n \in \mathbb{N}^* \tag{1}$$

Η παρατήρηση (8.2.4) μας πληροφορεί ότι ο $N = p_1 \cdot p_2 \cdot ... \cdot p_n + 1$ έχει πρώτο διαιρέτη στο διάστημα $(p_n,\ N)$ ή είναι ο ίδιος πρώτος. Έστω $j = 1$, τότε έχουμε ότι:

$$p_{n+1} \leq \left(\prod_{i=1}^{n} p_i\right) + 1,\quad \forall n \in \mathbb{N}^* \tag{2}$$

Εφαρμόζοντας μαθηματική επαγωγή:

- Για $n\ =\ 1$: $(1) \Rightarrow p_1 < 2^{2^1} \Leftrightarrow 2 < 4$, ισχύει άρα η (2) αληθεύει.
- Έστω ότι η (1) αληθεύει όλα τα $2 \leq k < n$, άρα:

$$p_k < 2^{2^k}\ \Rightarrow\ \prod_{i=1}^{k} p_i < \prod_{i=1}^{k} 2^{2^i} \tag{3}$$

- Εξετάζουμε αν η (1) αληθεύει για $n\ =\ k+1$:

$$(2)\ \Rightarrow\ p_{k+1} \leq \left(\prod_{i=1}^{k} p_i\right) + 1\ \xRightarrow{(3)}\ p_{k+1} < \left(\prod_{i=1}^{k} 2^{2^i}\right) + 1 \tag{4}$$

$$\prod_{i=1}^{k} 2^{2^i} = 2^2 \cdot 2^4 \cdot 2^8 \cdot ... \cdot 2^{2^k} = 2^{2+4+8+...+2^k} = 2^{\sum_{i=1}^{k} 2^i} \tag{5}$$

Από την ταυτότητα:

$$a^m - b^m \; = \; (a-b)(a^{m-1} + a^{m-2}b + a^{m-3}b^2 + ... + ab^{m-2} + b^{m-1})$$

για $a = 2,\ b = 1,\ m = k+1$:

$$\begin{aligned} &2^{k+1} - 1^{k+1} = (2-1)(2^k + 2^{k-1} \cdot 1 + 2^{k-2} \cdot 1^2 + ... + 2 \cdot 1^{k-1} + 1^k) \Leftrightarrow \\ &\Leftrightarrow 2^{k+1} - 1 = 2^k + 2^{k-1} + 2^{k-2} + ... + 2 + 1 \Leftrightarrow \\ &\Leftrightarrow 2^{k+1} - 2 = 2^k + 2^{k-1} + 2^{k-2} + ... + 2 \end{aligned}$$

Άρα

$$\sum_{i=1}^{k} 2^i \; = \; 2^{k+1} - 2 \tag{6}$$

Από τις σχέσεις (5) και (6) :

$$\prod_{i=1}^{k} 2^{2^i} = 2^{2^{k+1}-2} \tag{7}$$

Από τις σχέσεις (4) και (7) :

$$p_{k+1} < 2^{2^{k+1}-2} + 1 \tag{8}$$

Λύνουμε την ανίσωση:

$$\begin{aligned} 2^{u-2} + 1 < 2^u &\Leftrightarrow \frac{2^u}{4} + 1 < 2^u \\ &\Leftrightarrow 2^u + 4 < 4 \cdot 2^u \\ &\Leftrightarrow 3 \cdot 2^u > 4 \end{aligned}$$

Η παραπάνω ανίσωση ισχύει για κάθε u θετικό ακέραιο (αληθεύει για $u = 1$ και $f(u) = 2^u$ γνησίως αύξουσα). Άρα, αν θέσουμε όπου u το 2^{k+1} τότε, έχουμε:

$$2^{2^{k+1}-2} + 1 < 2^{2^{k+1}}, \;\; \forall k \in \mathbb{N}^* \tag{9}$$

Από τις σχέσεις (8) και (9), προκύπτει ότι αληθεύει η (1) για $n \; = \; k+1$:

$$p_{k+1} < 2^{2^{k+1}}$$

Τελικά, αποδείξαμε τη σχέση (1):

$$p_n < 2^{2^n}, \;\; \forall n \in \mathbb{N}^*$$

Άσκηση 4 (8.32)

- *Ερώτημα (i):*

$$\begin{aligned} gcd(a,\ b)\ &=\ 1 \Leftrightarrow \exists\, n_1, m_1 \in \mathbb{Z}:\ an_1 + bm_1 = 1 && (1)\\ d\ &\coloneqq\ gcd(c,\ b) \Leftrightarrow \exists\, n_2, m_2 \in \mathbb{Z}:\ cn_2 + bm_2 = d && (2)\\ d'\ &\coloneqq\ gcd(ac,\ b) \Leftrightarrow \exists\, n_3, m_3 \in \mathbb{Z}:\ acn_3 + bm_3 = d' && (3) \end{aligned}$$

Θα αποδειχθεί ότι $d|d'$ και $d'|d$, άρα είναι ίσα.

$$\begin{aligned} (1) &\Leftrightarrow an_1d + bm_1d = d \overset{(2)}{\Longleftrightarrow} an_1(cn_2 + bm_2) + bm_1(cn_2 + bm_2) = d \Leftrightarrow\\ &\Leftrightarrow acn_1n_2 + abn_1m_2 + bcn_2m_1 + b^2m_1m_2 = d \Leftrightarrow\\ &\Leftrightarrow n_1n_2 \cdot ac + (an_1m_2 + cn_2m_1 + bm_1m_2)b = d && (4) \end{aligned}$$

Άρα d είναι γραμμικός συνδυασμός των $a \cdot c$ και b. Έχουμε ότι d' ο ΜΚΔ των $a \cdot c$ και b, άρα:

$$\left.\begin{aligned} d'&|ac\\ d'&|b \end{aligned}\right\} \Rightarrow d'|(u \cdot ac + v \cdot b),\ \forall u, v \in \mathbb{Z}. \qquad (5)$$

Επομένως, $(4) \wedge (5) \Rightarrow d'|d$. Επίσης d ο ΜΚΔ των c και b:

$$\left.\begin{aligned} d&|c\\ d&|b \end{aligned}\right\} \Rightarrow d|(u \cdot c + v \cdot b),\ \forall u, v \in \mathbb{Z} \xRightarrow[u=an_3, v=m_3]{(3)} d|d'$$

Τελικά:

$$\left.\begin{aligned} &d|d'\\ &d'|d\\ &d, d' \geq 0 \text{ ως ΜΚΔ} \end{aligned}\right\} \Rightarrow d' = d \Leftrightarrow gcd(ac,\ b) = gcd(c,\ b)$$

- *Ερώτημα (ii):*

Έστω $d\ \coloneqq\ gcd(a + b,\ a - b)$, τότε:

$$d \,|\, [(a + b)n + (a - b)m],\ \forall n, m \in \mathbb{Z} \Rightarrow \begin{cases} d|2a & \text{για } (n,\ m) = (1,\ 1)\\ d|2b & \text{για } (n,\ m) = (1,\ -1) \end{cases}$$

Άρα:

$$d \,|\, gcd(2a,\ 2b) \Leftrightarrow d \,|\, [2 \cdot gcd(a, b)] \Leftrightarrow d \,|\, 2 \Leftrightarrow (d = \pm 1 \vee d = \pm 2)$$

Ισχύει ότι $d \geq 0$ ως ΜΚΔ, άρα $d \in \{1.\ 2\}$. Συγκεκριμένα, αν a και b περιττοί ακέραιοι τότε $a + b$ και $a - b$ άρτιοι, άρα $d = 2$.

$$\begin{cases} a \coloneqq 2k_1 + 1\\ b \coloneqq 2k_2 + 1 \end{cases} \Rightarrow \begin{cases} a + b = 2k_1 + 2k_2 + 2 = 2(k_1 + k_2 + 1)\\ a - b = 2k_1 - 2k_2 = 2(k_1 - k_2) \end{cases} \xRightarrow[c_2 = k_1 - k_2]{c_1 = k_1 + k_2 + 1} \begin{cases} a + b = 2c_1\\ a - b = 2c_2 \end{cases}$$

$$d = gcd(a + b,\ a - b) = gcd(2c_1,\ 2c_2) = 2 \cdot gcd(c_1,\ c_2) \neq 1 \xRightarrow{d \in \{1,\ 2\}} d = 2$$

- *Ερώτημα (iii):*

  Ισχύουν οι παρακάτω προτάσεις:

$$\mathbf{a \equiv b\,(mod\,m) \wedge b \equiv c\,(mod\,m) \Rightarrow a \equiv c\,(mod\,m)} \tag{Π.1}$$

$$\mathbf{a \equiv b\,(mod\,m) \Rightarrow a^n \equiv b^n\,(mod\,m)} \tag{Π.2}$$

$$\mathbf{a \equiv b\,(mod\,m) \Rightarrow na \equiv nb\,(mod\,m)} \tag{Π.3}$$

  Αρχικά:

$$gcd(a,\, b) = 1 \Rightarrow \exists\, x, y \in \mathbb{Z}:\ 1 = ax + by \tag{6}$$

  Έστω $d := gcd(2^a - 1,\, 2^b - 1)$, τότε:

$$\begin{cases} \exists\, g_1 \in \mathbb{Z}:\ 2^a - 1 = g_1 d \\ \exists\, g_2 \in \mathbb{Z}:\ 2^b - 1 = g_2 d \end{cases} \Leftrightarrow \begin{cases} 2^a = g_1 d + 1 \\ 2^b = g_2 d + 1 \end{cases}$$

  Άρα:

$$\begin{cases} 2^a \equiv 1\,(mod\,d) \\ 2^b \equiv 1\,(mod\,d) \end{cases} \xRightarrow{\mathbf{(Π.2)}} \begin{cases} (2^a)^x \equiv 1^x\,(mod\,d) \\ (2^b)^y \equiv 1^y\,(mod\,d) \end{cases}$$

  ή αλλιώς:

$$(2^a)^x \equiv 1\,(mod\,d) \tag{7}$$

$$(2^b)^y \equiv 1\,(mod\,d) \tag{8}$$

  Από την (7) και την (Π.3):

$$(2^a)^x (2^b)^y \equiv (2^b)^y\,(mod\,d) \tag{9}$$

  Από τις (8), (9) και (Π.1):

$$\mathbf{(2^a)^x (2^b)^y \equiv 1\,(mod\,d)} \tag{10}$$

  Τελικά, από (6) και (10):

$$\begin{aligned} &2 = 2^{ax+by} = (2^a)^x (2^b)^y \equiv 1\,(mod\,d) \Rightarrow \\ &\Rightarrow 2 \equiv 1\,(mod\,d) \Rightarrow \\ &\Rightarrow d | 2 - 1 \Rightarrow d | 1 \Rightarrow \\ &\Rightarrow d = \pm 1 \xRightarrow{d > 0} d = 1 \end{aligned}$$

  Αποδείχτηκε ότι για $gcd(a,\, b) = 1$:

$$\mathbf{gcd(2^a - 1,\, 2^b - 1) = 1}$$

- *Ερώτημα (iv):*

  Οι διαιρέτες του $p \in \mathbb{P}$ είναι οι 1 και p, ενώ οι διαιρέτες του $q \in \mathbb{P}$ είναι οι 1 και q. Συνεπώς $\mathbf{gcd(p,\, q) = 1}$ (αφού $p \neq q$) και άρα $\mathbf{gcd(2^p - 1,\, 2^q - 1) = 1}$, καθώς αποδείχτηκε στο προηγούμενο ερώτημα.

Άσκηση 6 (8.45)

Το Python script **"ex6.py"** ελέγχει αν η ανίσωση

$$\frac{\sigma(n)}{n} < \frac{e^\gamma}{2} \cdot ln(ln(n)) + \frac{0.74}{ln(ln(n))} \tag{6.1}$$

αληθεύει για κάθε n περιττό (θετικό) ακέραιο με $n < 2^{20}$. Εξαίρεση αποτελεί η τιμή $n = 1$ καθώς το δεξί μέλος της ανίσωσης βγαίνει εκτός του πεδίου ορισμού του. Επομένως ελέγχεται η τιμή αληθείας της παραπάνω σχέσης για όλα τα

$$n \in \{3,\ 5,\ 7,\ 9,\ ...,\ 2^{20} - 3,\ 2^{20} - 1\}$$

Όλα βασίζονται στη συνάρτηση `find_counter_argument_in_interval(...)`, της οποίας η μαθηματική μοντελοποίηση θα ήταν

$$f:\ \mathbb{Z}^2 \to \{False,\ True\} \times \mathbb{Z},\ \text{με}$$

$$f(a,\ b) = \begin{cases} (False,\ -1) & \text{αν ισχύει η (6.1) για κάθε } a \le n < b,\ \ 2 \not| n \\ (True,\ n_0) & \text{αν } \exists n_0 :\ \frac{\sigma(n_0)}{n_0} \ge \frac{e^\gamma}{2} \cdot ln(ln(n_0)) + \frac{0.74}{ln(ln(n_0))},\ a \le n_0 < b,\ \ 2 \not| n_0 \end{cases}$$

Τεχνάσματα για τη βελτιστοποίηση της παραπάνω συνάρτησης αποτελούν ο εκ των πρωτέρον υπολογισμός της σταθεράς $e^\gamma/2$ και η αποθήκευση της έκφρασης $ln(ln(n))$ σε μεταβλητή για την αποφυγή των 2 επιπλέων κλήσεων της συνάρτησης `math.log()`.

Κατά τ' άλλα, αυτές οι μικρές βελτιστοποιήσεις δεν είναι ιδιαίτερα αποδοτικές για $a = 3$ και $b = 2^{20}$. Επομένως, χρησιμοποιήθηκε παραλληλία σε επίπεδο hardware χρησιμοποιώντας τη built-in βιβλιοθήκη `concurrent.futures` της Python.

Αντί να υπολογιστεί το $f(3,\ 2^{20})$ επιλέγονται 9 τιμές $v_1 < v_2 < ... < v_9$ ώστε να γίνει ένας διαμερισμός του διαστήματος $[3,\ 2^{20})$, δηλαδή (υποθέτοντας $v_0 = 3$ και $v_{10} = 2^{20}$):

$$\bigcup_{i=0}^{9} [v_i,\ v_{i+1})\ =\ [v_0,\ v_{10})\ =\ [3,\ 2^{20})$$

$$[v_i,\ v_{i+1}) \cap [v_j,\ v_{j+1}) = \emptyset,\ \ \forall i \ne j$$

Επιπλέον, υποθέτοντας ότι f_1 η λογική μεταβλητή που επιστρέφεται από την f και f_2, αντίστοιχα, ο ακέραιος, τότε:

$$\sum_{i=0}^{9} f_1(v_i,\ v_{i+1})\ =\ f_1(3,\ 2^{20})$$

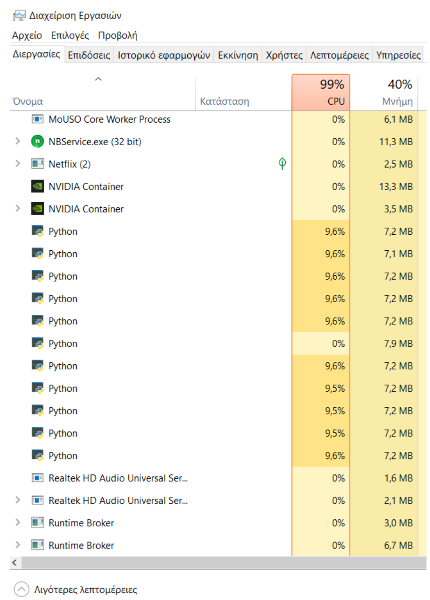
(**άθροισμα:** εννοώντας λογική διάζευξη | Το f_2 εκτυπώνεται ανν $f_1 = True$)

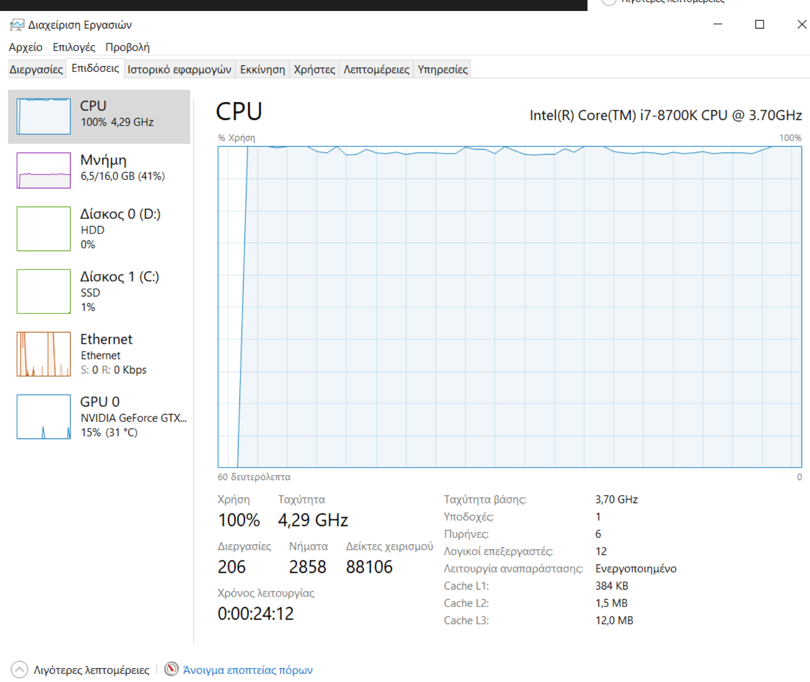
Άρα, μπορούμε να διαπιστώσουμε αν η (6.1) αληθεύει για κάθε $v_0 \le n < v_{10}$ ελέγχοντας κάθε ένα από τα 10 διαδοχικά ζεύγη παράλληλα δημιουργώντας ένα `ProcessPoolExecutor` με 10 παράλληλες διαδικασίες, μία για κάθε $f(v_i,\ v_{i+1})$. Αν έστω και σε ένα από αυτά τα διαστήματα βρεθεί αντιπαράδειγμα τότε $\sum_{i=0}^{9} f_1(v_i,\ v_{i+1})\ =\ True$ και το πρόγραμμα θα τερματίσει με το μήνυμα `"Mathematical formula is invalid"` αλλιώς με `"Program finished successfully"`.

Σχολιασμός Αποτελεσμάτων:

Το πρόγραμμα χρειάστηκε 20 λεπτά για να αποφανθεί ότι **δεν υπάρχει** περιττός ακέραιος $3 \leq n < 2^{20}$ που να παραβιάζει την ανίσωση (6.1), δηλαδή όσο και η πιο αργοπορημένη διαδικασία. Η σειριακή εκδοχή του προγράμματος θα χρειαζόταν περίπου 2 ώρες και 16 λεπτά (περίπου 6.76 φορές περισσότερο), όσο το άθροισμα των χρόνων εκτέλεσης των 10 διαδικασιών. Ιδανικά, αν είχαν επιλεχθεί τα βέλτιστα v_i ως άκρα των διαστημάτων, τότε όλες οι διαδικασίες θα έκαναν τον ίδιο χρόνο, άρα το πρόγραμμα θα ήταν 10 φορές πιο γρήγορο. Οι τιμές των v_i επιλέχθηκαν ενστικτωδώς και δεν έγινε κάποια διαδικασία εμβάθυνσης στο παραπάνω θέμα. Έτσι, αποδείχθηκε το ζητούμενο της άσκησης.

```
Available CPU cores:  12
> Process " 1" finished after   386.89 sec
> Process " 2" finished after   590.76 sec
> Process " 3" finished after   705.81 sec
> Process "10" finished after   719.31 sec
> Process " 5" finished after   785.30 sec
> Process " 4" finished after   793.24 sec
> Process " 9" finished after   913.02 sec
> Process " 6" finished after   926.74 sec
> Process " 7" finished after  1166.04 sec
> Process " 8" finished after  1212.88 sec
Program finished successfully
Elapsed time:  1213.08 second(s)
```





Άσκηση 7 (9.18)

Έστω η συνάρτηση $\mathcal{P}(n) = (\vec{p},\ \vec{\alpha})$, όπου $\vec{p} \in \mathbb{P}^k$ και $\vec{\alpha} \in \mathbb{N}^{*k}$, $k \in \mathbb{N}^*$ εκφράζουν την παραγοντοποίηση του n σε πρώτους, δηλαδή:

$$n = \prod_{i=1}^{k} p_i^{\alpha_i}$$
$$p_i \neq p_j \ \forall i \neq j$$
$$p_i \in \mathbb{P},\quad \alpha_i \in \mathbb{N}^*$$

Αυτό ακριβώς υλοποιεί η συνάρτηση `prime_factorization(n:int)` (αλγόριθμος δοκιμαστικής διαίρεσης). Προκύπτει γρήγορα απάντηση για τους ακεραίους 6553130926752006031481761 και 9999109081 καθώς όλοι οι πρώτοι παράγοντές τους είναι μικροί.

Σύμφωνα με το κριτήριο Korselt, για κάθε έναν από αυτούς τους πρέπει να ισχύει:

- $a_i = 1$ για κάθε $1 \leq i \leq k$
- $p_i - 1 \mid n - 1$ για κάθε $1 \leq i \leq k$

Οι δοσμένοι ακέραιοι πληρούν τα παραπάνω κριτήρια.

(Υλοποίηση: **κώδικας/ex7/ex7.py**)

Άσκηση 8 (9.28)

Οι ακέραιοι $835335 \cdot 2^{39014} \pm 1$ περνούν το τεστ του Fermat.

- $\mathbf{n_1 = 835335 \cdot 2^{39014} + 1}$: Ψευδοπρώτος ως προς τη βάση $a_1 \in \mathbb{Z},\ \ a_1 \approx 2^{39033.011512}$
- $\mathbf{n_2 = 835335 \cdot 2^{39014} - 1}$: Ψευδοπρώτος ως προς τη βάση $a_2 \in \mathbb{Z},\ \ a_2 \approx 2^{39033.632355}$

Οι a_1 και a_2 είναι τυπωμένοι αναλυτικά στα αρχεία `n1.txt` και `n2.txt` αντίστοιχα.

```
n1 is probable prime with respect 2^39033.011512 (elapsed: 84.548 sec)
n2 is probable prime with respect 2^39033.632355 (elapsed: 166.835 sec)
```

(Υλοποίηση: **κώδικας/ex8/ex8.py**)

Άσκηση 9 (10.1)

Ο αλγόριθμος δοκιμαστικής διαίρεσης εξάγει τα παρακάτω αποτελέσματα:

$$\mathbf{2^{62} - 1} = 3 \cdot 715827883 \cdot 2147483647$$

$$\mathbf{2^{102} - 1} = 3^2 \cdot 7 \cdot 103 \cdot 307 \cdot 2143 \cdot 2857 \cdot 6529 \cdot 11119 \cdot 43691 \cdot 131071$$

(Υλοποίηση: **κώδικας/ex9/ex9.py**)

Άσκηση 10 (10.8)

Το πρόγραμμα επιλέγει 1000 τυχαίους ακεραίους των 100 bits, δηλαδή:

$$n_i \xleftarrow{\$} [2^{99},\ 2^{100}) \cap \mathbb{Z},\ \ i \in \{1,\ 2,\ ...,\ 1000\}$$

Η συνάρτηση `lehman(int, float)` έχει δυο παραμέτρους εισόδου: τον αριθμό προς παραγοντοποιηση, **n** και ένα χρονικό όριο σε δευτερόλεπτα (ίσο με 10 στη συγκεκριμένη άσκηση). Ως αποτέλεσμα επιστρέφει έναν παράγοντα **f** του n. Ο αλγόριθμος είναι επιτυχής άν

1. τερματίσει πριν το χρονικό όριο και επιπλέον
2. $f \neq 1 \ \vee \ f \neq n \ \vee \ f \in \mathbb{Z}$ (δεν έχει τιμή None)

Συνολικά, βρέθηκε παράγοντας για 16 ακεραίους. Ενδεικτικά:

- $n_{21} = 674.902.139.001.917.536.149.940.578.006$ με $1.739.779.667.412.498 \mid n_{21}$
- $n_{31} = 876.328.564.129.523.250.183.808.043.827$ με $29.891.192.611.171 \mid n_{31}$
- $n_{551} = 949.629.530.912.133.951.962.690.604.832$ με $49.596.489.117.889.648 \mid n_{551}$

Τα αποτελέσματα αναγράφονται αναλυτικά στο αρχείο **"results.txt"**.

```
Produced a factor for   1.600% of integers (16 / 1000)
Elapsed time: 1641.572 sec
```

(Υλοποίηση: **κώδικας/ex10/ex10.py**)

Άσκηση 11 (10.21)

Η υλοποίηση που έγινε για τον Pollard-ρ στη συγκεκριμένη άσκηση στοχεύει στην εύρεση ενός παράγοντα του ακεραίου N (μοναδική παράμετρος). Ισχύουν οι αρχικοποιήσεις:

- $F(x) = (x^2 + 1)\, mod\, N$
- $X_0 \xleftarrow{\$} \{2,\ 3,\ ...,\ N-1\}$
- $X \coloneqq X_0$
- $Y \coloneqq X_0$

Ο αλγόριθμος τερματίζει όταν $1 < d < N$ και ο d επιστρέφεται ως αποτέλεσμα, όπου
$d = gcd(|X - Y|,\ N)$.

Για $\mathbf{N = 2^{257} - 1}$ και αρχικοποιώντας το σπόρο της γεννήτριας της Python με $s = 42$, λαμβάνονται τα ακόλουθα αποτελέσματα:

```
Found non-trivial factor: 535006138814359
Elapsed time: 70.564 sec
Execution steps: 17571888
```

(Υλοποίηση: **κώδικας/ex11/ex11.py**)

Άσκηση 12 (11.3)

Για τη λύση της άσκησης χρησιμοποιήθηκαν:

1. **Συνάρτηση Δοκιμαστικής Διαίρεσης:** Όπως και στις προηγούμενες ασκήσεις.
2. **Συνάρτηση Euler:** Για $n = p_1^{a_1} \cdot p_2^{a_2} \cdot ... \cdot p_k^{a_k}$, επιστρέφει:

$$\phi(n) = n \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$$

   ή ισοδύναμα, για την αποφυγή χρήσης αριθμών κινητής υποδιαστολής:

$$\phi(n) = \Phi_{k+1}(n)$$

$$\Phi_i(n) = \begin{cases} \Phi_{i-1}(n) - \lfloor \frac{\Phi_{i-1}(n)}{p_{i-1}} \rfloor & \text{για } i \geq 2 \\ n & \text{για } i = 1 \end{cases}$$

3. **Αλγόριθμος Υπολογισμού Ιδιωτικού Κλειδιού:** Για είσοδο $pk = (e,\ N)$, επιστρέφει $sk = (d,\ N)$ με $e \cdot d \equiv 1\ (mod\, \phi(N))$.
4. **Αλγόριθμος RSA:** Για είσοδο το κρυπτοκείμενο και το sk, επιστρέφει το αποκρυπτογραφημένο μήνυμα.

Αρχικά, έχουμε $pk = (19,\ 11413)$ και άρα υπολογίζουμε $sk = (1179,\ 11413)$. Έτσι, για:

$$C = (3203, 909, 3143, 5255, 5343, 3203, 909, 9958, 5278, 5343, 9958, 5278,\\ 4674, 909, 9958, 792, 909, 4132, 3143, 9958, 3203, 5343, 792, 3143, 4443)$$

τότε η κλήση της συνάρτησης `RSA(sk, C)` επιστρέφει:

$$M = (119, 101, 108, 99, 111, 119, 101, 32, 116, 111, 32, 116,\\ 104, 101, 32, 114, 101, 97, 108, 32, 119, 111, 114, 108, 100)$$

Του οποίου η αποκωδικοποίηση σε χαρακτήρες ASCII είναι:

"welcowe to the real world"

(Υλοποίηση: **κώδικας/ex12/ex12.py**)

Άσκηση 13 (12.2)

Για είσοδο $pk = (e,\ N) = (50736902528669041,\ 194749497518847283)$ δίνονται τα βήματα της επίθεσης του Wiener:

1. Αποθηκεύουμε σε μια λίστα, A, το συνεχές κλάσμα του $\frac{e}{N}$ με ακρίβεια 40 συντελεστών:

$$\frac{e}{N} = [0;\ 3, 1, 5, 5, 3, 2, 1, 1, 2, 1, 2, 1, 1, 1, 2, 1, 4, 1, 26, 4,\\ 2, 3, 1, 18, 10, 6, 3, 180, 2, 2, 1, 1, 4, 2, 5, 1, 2, 3, 83, 9]$$

2. Για κάθε $i \in [1,\ 40]$ αποθηκεύουμε σε μια λίστα, F, τους πραγματικούς αριθμούς x_i, όπου $x_i = [A_0;\ A_1, ..., A_i]$

3. Για κάθε $i \in [1,\ 40]$ χρησιμοποιούμε την κλάση `fractions.Fraction` για να μας επιστρέφει τους ακεραίους N_i και D_i, όπου $\frac{N_i}{D_i}$ το ανάγωγο κλάσμα του x_i. Στην επανάληψη για $i = 12$ ισχύει $x_{12} \approx 0.260523921268139$, άρα $(N_{12},\ D_{12}) = (5440,\ 20881)$.

4. Ισχύει ότι $\mathbf{2^{e \cdot D_{12}} \equiv 2 (\mathrm{mod}\ N)}$, επομένως επιστρέφεται το $D_{12} = 20881$ ως πιθανό ιδιωτικό κλειδί.

Εισάγοντας το κρυπτοκείμενο σε έναν αποκωδικοποιητή Base64 προκύπτει μια λίστα Python:

$C = [47406263192693509, 51065178201172223, ..., 134434295894803806, 57208077766585306]$

Εισάγοντας το $sk = (D_{12},\ N)$ και το C στον RSA προκύπτει μια λίστα ακεραίων των οποίων η κωδικοποίηση σε χαρακτήρες ASCII είναι:

" **Just because you are a character doesn't mean that you have character**"

(Υλοποίηση: **κώδικας/ex13/ex13.py**)

Άσκηση 14 (13.2)

Για $N = 899,\ e = 839,\ m = 3,\ s = 301 \ \Rightarrow\ \mathbf{a = s^e \bmod N = 675} \neq m$, άρα η ψηφιακή υπογραφή s είναι λάθος.

(Υλοποίηση: **κώδικας/ex14/ex14.py**)