

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Πληροφορικής

Τεχνική αναφορά για NGE-06-03

Κρυπτογραφία 101

Αλέξανδρος Κόρκος
alexkork@csd.auth.gr
3870

Θεσσαλονίκη, 30 Ιουλίου 2023



Το έργο αυτό διατίθεται υπό τους όρους της άδειας **Create Commons "Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Παρόμοια Διανομή 4.0 Διεθνές"**.

Περιεχόμενα

I	Συμμετρική Κρυπτογραφία	5
1	Σύστημα μετατόπισης	5
1.1	Εκφώνηση	5
1.2	Λύση	5
1.2.1	Μαθηματική επίλυση	5
1.2.2	Λεπτομέρειες για την υλοποίηση	5
1.2.3	Συμπέρασμα	6
2	Το σύστημα του Vigenere	6
2.1	Εκφώνηση	6
2.2	Λύση	6
2.2.1	Λεπτομέρειες για την υλοποίηση	6
2.2.2	Συμπέρασμα	6
3	Κυκλική κύλιση	7
3.1	Εκφώνηση	7
3.2	Λύση	7
3.2.1	Μαθηματική επίλυση	7
3.2.2	Λεπτομέρειες για την υλοποίηση	8
3.2.3	Συμπέρασμα	8
4	Τέλεια ασφάλεια	8
4.1	Εκφώνηση	8
4.2	Λύση	9
4.2.1	Μαθηματική επίλυση	9
4.2.2	Συμπέρασμα	9
5	One Time Pad	9
5.1	Εκφώνηση	9
5.2	Λύση	9
5.2.1	Λεπτομέρειες για την υλοποίηση	9
5.2.2	Συμπέρασμα	10
6	Η αριθμοθεωρητική συνάρτηση Möbius	10
6.1	Εκφώνηση	10
6.2	Λύση	10
6.2.1	Λεπτομέρειες για την υλοποίηση	10
6.2.2	Συμπέρασμα	10
7	Rivest Cipher 4 (RC4)	11
7.1	Εκφώνηση	11
7.2	Λύση	11
7.2.1	Λεπτομέρειες για την υλοποίηση	11
7.2.2	Συμπέρασμα	11
8	Διαφορική ομοιομορφία	11
8.1	Εκφώνηση	11
8.2	Λύση	12
8.2.1	Λεπτομέρειες για την υλοποίηση	12
8.2.2	Συμπέρασμα	12

9	Avalanche effect	13
9.1	Εκφώνηση	13
9.2	Λύση	13
9.2.1	Λεπτομέρειες για την υλοποίηση	13
9.2.2	Συμπέρασμα	13
10	Capture The Flag	13
10.1	Εκφώνηση	13
10.2	Λύση	13
10.3	Πορεία	14
II	Κρυπτογραφία Δημοσίου Κλειδιού	14
11	Diffie-Hellman	14
11.1	Εκφώνηση	14
11.2	Λύση	14
11.2.1	Λεπτομέρειες για την υλοποίηση	14
11.2.2	Συμπέρασμα	14
12	Γρήγορη ύψωση σε δύναμή	14
12.1	Εκφώνηση	14
12.2	Λύση	15
12.2.1	Λεπτομέρειες για την υλοποίηση	15
12.2.2	Συμπέρασμα	15
13	Πρώτοι αριθμοί	15
13.1	Εκφώνηση	15
13.2	Λύση	15
13.2.1	Μαθηματική επίλυση	15
14	Ιδιότητες του gcd	15
14.1	Εκφώνηση	15
14.2	Λύση	16
14.2.1	Μαθηματική επίλυση	16
15	Ακόμα ένα πρόβλημα με gcd	17
15.1	Εκφώνηση	17
15.2	Λύση	17
15.2.1	Μαθηματική επίλυση	17
16	Ιδιότητα των περιττών ακέραιων	18
16.1	Εκφώνηση	18
16.2	Λύση	18
16.2.1	Λεπτομέρειες για την υλοποίηση	18
16.2.2	Συμπέρασμα	19
17	Carmichael	19
17.1	Εκφώνηση	19
17.2	Λύση	19
17.2.1	Λεπτομέρειες για την υλοποίηση	19
17.2.2	Συμπέρασμα	19

18 Τεστ του Fermat	19
18.1 Εκφώνηση	19
18.2 Λύση	20
18.2.1 Συμπέρασμα	20
19 Trial Division	20
19.1 Εκφώνηση	20
19.2 Λύση	20
19.2.1 Λεπτομέρειες για την υλοποίηση	20
19.2.2 Συμπέρασμα	20
20 Αλγόριθμος του Lehman	20
20.1 Εκφώνηση	20
20.2 Λύση	21
20.2.1 Συμπέρασμα	21
21 Αλγόριθμος του Pollard	21
21.1 Εκφώνηση	21
21.2 Λύση	21
21.2.1 Λεπτομέρειες για την υλοποίηση	21
21.2.2 Συμπέρασμα	21
22 RSA	22
22.1 Εκφώνηση	22
22.2 Λύση	22
22.2.1 Λεπτομέρειες για την υλοποίηση	22
22.2.2 Συμπέρασμα	22
23 Wiener Attack	23
23.1 Εκφώνηση	23
23.2 Λύση	23
23.2.1 Λεπτομέρειες για την υλοποίηση	23
23.2.2 Συμπέρασμα	23
24 Naive RSA	23
24.1 Εκφώνηση	23
24.2 Λύση	23
24.2.1 Λεπτομέρειες για την υλοποίηση	23
24.2.2 Συμπέρασμα	24
25 gpg	24
25.1 Εκφώνηση	24
25.2 Λύση	24
25.2.1 Συμπέρασμα	24

Μέρος Ι

Συμμετρική Κρυπτογραφία

1 Σύστημα μετατόπισης

1.1 Εκφώνηση

Το επόμενο κρυπτόγραμμα έχει ληφθεί:

οκηθμφδζθγοθχυκχσφθμφμχγ

Ο αλγόριθμος κρυπτογράφησης είναι ο εξής:

Κάθε γράμμα του αρχικού μας μηνύματος αντικαθίσταται από την αριθμητική του τιμή (α: 1, ..., ω: 24). Ας είναι x_0 μία ρίζα του τριωνύμου $g(x) = x^2 + 3x + 1$. Σε κάθε αριθμό του μηνυμάτος μου προσθέτω την τιμή του πολυωνύμου $f(x) = x^5 + 3x^4 + 3x^3 + 7x^2 + 5x + 4$, στο x_0 . Αντικαθιστώ κάθε αριθμό με το αντίστοιχο γράμμα. Βρείτε το αρχικό μήνυμα.

1.2 Λύση

1.2.1 Μαθηματική επίλυση

Από την εκφώνηση, γνωρίζω ότι $g(x_0) = 0$.

Επίσης, θα απλοποιήσω την $f(x)$ μέσω της διαίρεση πολυωνυμικών συναρτήσεων έτσι ώστε να εκμεταλλευτώ την παραπάνω πληροφορία. Συνεπώς:

$$f(x) = g(x) \cdot h(x) \Rightarrow h(x) = \frac{f(x)}{g(x)} \Rightarrow h(x) = x^3 + 2x + 1 + \frac{3}{x^2 + 3x + 1} \quad (1)$$

Κάνοντας εφαρμογή της επιμεριστικής ιδιότητας στην (1), προκύπτει:

$$f(x) = (x^3 + 2x + 1) \cdot g(x) + \frac{3}{x^2 + 3x + 1} \cdot \cancel{g(x)} \Rightarrow f(x) = (x^3 + 2x + 1) \cdot g(x) + 3 \quad (2)$$

Εξετάζω την συνάρτηση (2) για $x = x_0$:

$$f(x_0) = (x_0^3 + 2x_0 + 1) \cdot \cancel{g(x_0)} + 3 \Rightarrow f(x_0) = 3 \quad (3)$$

Άρα, όλα τα γράμματα της αλφάβητα έχουν μετατοπιστεί κατά 3 θέσεις.

1.2.2 Λεπτομέρειες για την υλοποίηση

Εκτέλεση αρχείου:

```
python >> python shift.py
```

Μια απλή υλοποίηση του συστήματος μετατόπισης, δηλαδή της συνάρτησης:

$$D_n(x) = x - n \pmod{24}, n = 3 \quad (4)$$

1.2.3 Συμπέρασμα

Εκτελώντας το αρχείο για το μηνύματος εκφώνησης επιστρέφεται το εξής αποκρυπτογραφημένο μήνυμα:

Μηδεις αγεωμετρητος εισιτω

όπου ήταν φράση που βρισκόταν στην είσοδο της ακαδημίας του Πλάτωνα ¹.

2 Το σύστημα του Vigenere

2.1 Εκφώνηση

Αποκρυπτογραφήστε το κείμενο [2], που κρυπτογραφήθηκε με τον αλγόριθμο του Vigenere.

Υποδ. Για την αποκρυπτογράφηση συστήνουμε να χρησιμοποιήσετε python. Για το μήκος του κλειδιού μπορείτε να χρησιμοποιήσετε είτε test Kasiski ή την μέθοδο του Friedman.

2.2 Λύση

2.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- functools
- re

```
python >> vegenere.py >> file_name
```

Ως όνομα αρχείου δίνεται το αρχείο που θέλουμε να αποκρυπτογραφήσουμε.

Αρχικά, η βασικότερη συνάρτηση είναι η kasiski, που υλοποιεί την μέθοδο, σύμφωνα με ². Για να επιτευχθεί η frequency analysis, υλοποιείται η συνάρτηση frequencyAnalysis που βασίζεται στο άρθρο [1].

2.2.2 Συμπέρασμα

Για να βρεθεί το αποκρυπτογραφημένο μήνυμα, διακρίνεται σε 3 βήματα που περιγράφονται στην συνέχεια.

Εκτελώντας λοιπόν το τεστ Kasiski, βρίσκετε πως το μήκος του κλειδιού είναι 7 χαρακτήρες. Για να φτάσουμε σε αυτό τον αριθμό, δοκιμάσθηκε η ομαδοποίηση του κειμένου με διαφορά πλήθους χαρακτήρων. Για ομαδοποίηση ≥ 5 χαρακτήρων καταλήγουμε στο μήκος κλειδιού = 7. Για < 5 καταλήγουμε σε κλειδιά μήκους είτε 1 είτε 2 χαρακτήρων, που δεν αποκρυπτογραφούν το μήνυμα.

Στην συνέχεια, γίνεται ανάλυση συχνότητας γλώσσας (frequency analysis) για μήκος κλειδιού = 7. Η λέξη κλειδί στην οποία καταλήγουμε είναι SHANNON.

Τέλος, κάνοντας αποκρυπτογράφηση με λέξη κλειδί SHANNON προκύπτει το παρακάτω απόσπασμα μηνύματος και συμπεραίνετε πως το κλειδί είναι ορθό μιας και το αποκρυπτογραφημένο μήνυμα απαρτίζεται από αγγλικές λέξεις.

A VERY SMALL PERCENTAGE OF THE POPULATION PRODUCES THE GREATEST PROPORTION OF THE IMPORTANT IDEAS THIS IS A KIN TO AN IDEA PRESENTED BY AN ENGLISH MATHEMATICIAN TURING THAT THE HUMAN BRAIN IS SOMETHING LIKE A PIECE OF URANIUM THE HUMAN BRAIN IF IT IS BELOW THE CRITICAL LAP AND YOU SHOOT ONE ...

¹Πηγή

²Μέθοδος Kasiski

3 Κυκλική κύλιση

3.1 Εκφώνηση

Έστω ένα μήνυμα m , 16-bits. Θεωρούμε την κυκλική κύλιση προς τα αριστερά $\ll \alpha$ κατά α bits. Έστω ότι m κωδικοποιείται στο c σύμφωνα με τον τύπο,

$$c = m \oplus (m \ll 6) \oplus (m \ll 10) \quad (5)$$

Βρείτε τον τύπο αποκωδικοποίησης. Δηλαδή, γράψτε το m ως συνάρτηση του c . Υλοποιήστε κατάλληλο κώδικα για να δείξετε ότι ο τύπος που φτιάξατε είναι σωστός.

3.2 Λύση

3.2.1 Μαθηματική επίλυση

Γνωρίζω πως:

$$m = (m_0 \ m_1 \ m_2 \ m_3 \ m_4 \ \dots \ m_{11} \ m_{12} \ m_{13} \ m_{14} \ m_{15})$$

$$m \ll 6 = (m_6 \ m_7 \ m_8 \ m_9 \ m_{10} \ \dots \ m_4 \ m_3 \ m_2 \ m_1 \ m_0)$$

$$m \ll 10 = (m_{10} \ m_{11} \ m_{12} \ m_{13} \ m_{14} \ \dots \ m_4 \ m_3 \ m_2 \ m_1 \ m_0)$$

συνεπώς, έτσι μπορώ να ορίσω τον "πίνακα" c όπου κάθε σειρά περιλαμβάνει μια πράξη XOR μεταξύ αυτών.

$$c = \begin{pmatrix} m_0 & m_1 & m_2 & m_3 & m_4 & \dots & m_{11} & m_{12} & m_{13} & m_{14} & m_{15} \\ m_6 & m_7 & m_8 & m_9 & m_{10} & \dots & m_4 & m_3 & m_2 & m_1 & m_0 \\ m_{10} & m_{11} & m_{12} & m_{13} & m_{14} & \dots & m_4 & m_3 & m_2 & m_1 & m_0 \end{pmatrix} \quad (6)$$

Για αποκωδικοποιήσουμε το αρχικό μήνυμα, θα πρέπει να φθάσω $c \rightarrow m$ μέσω κάποιων πράξεων XOR και Left - Shift. Αρχικά, θα γίνει $c \oplus c \ll 10$ ε.ω. να απαλείψω την πρώτη και τρίτη "σειρά" του c καθώς θα είναι κοινές και στους δυο "πίνακες".

$$c \ll 10 = \begin{pmatrix} m_{10} & m_{11} & m_{12} & m_{13} & m_{14} & \dots & m_5 & m_6 & m_7 & m_8 & m_9 \\ m_0 & m_1 & m_2 & m_3 & m_4 & \dots & m_{11} & m_{12} & m_{13} & m_{14} & m_{15} \\ m_4 & m_5 & m_6 & m_7 & m_8 & \dots & m_{15} & m_0 & m_1 & m_2 & m_3 \end{pmatrix}$$

$$c_1 = c \oplus (c \ll 10) \quad (7)$$

$$(7) \Rightarrow c_1 = \begin{pmatrix} m_6 & m_7 & m_8 & m_9 & m_{10} & \dots & m_4 & m_3 & m_2 & m_1 & m_0 \\ m_4 & m_5 & m_6 & m_7 & m_8 & \dots & m_{15} & m_0 & m_1 & m_2 & m_3 \end{pmatrix}$$

Θα απαλείψω στην συνέχεια, την πρώτη σειρά του c_1 κάνοντάς XOR με τον c_1 μετατοπισμένο κατά 2.

$$c_1 \ll 2 = \begin{pmatrix} m_8 & m_9 & m_{10} & m_{11} & m_{12} & \dots & m_3 & m_4 & m_5 & m_6 & m_7 \\ m_6 & m_7 & m_8 & m_9 & m_{10} & \dots & m_4 & m_3 & m_2 & m_1 & m_0 \end{pmatrix}$$

$$c_2 = c_1 \oplus (c_1 \ll 2) \quad (8)$$

$$(8) \Rightarrow c_2 = \begin{pmatrix} m_8 & m_9 & m_{10} & m_{11} & m_{12} & \dots & m_3 & m_4 & m_5 & m_6 & m_7 \\ m_4 & m_5 & m_6 & m_7 & m_8 & \dots & m_{15} & m_0 & m_1 & m_2 & m_3 \end{pmatrix}$$

Έχοντας τώρα στον "πίνακα" c_2 2 σειρές, μπορώ μια μια μετατοπισμένη έκδοση του c να τον φέρω σε μοναδιαία σειρά.

$$c \ll 14 = \begin{pmatrix} m_{14} & m_{15} & m_0 & m_1 & m_2 & \dots & m_9 & m_{10} & m_{11} & m_{12} & m_{13} \\ m_8 & m_9 & m_{10} & m_{11} & m_{12} & \dots & m_3 & m_4 & m_5 & m_6 & m_7 \\ m_4 & m_5 & m_6 & m_7 & m_8 & \dots & m_{15} & m_0 & m_1 & m_2 & m_3 \end{pmatrix}$$

$$c_3 = c_2 \oplus (c \ll 14) \quad (9)$$

$$(9) \Rightarrow c_3 = (m_{14} \ m_{15} \ m_0 \ m_1 \ m_2 \ \dots \ m_9 \ m_{10} \ m_{11} \ m_{12} \ m_{13})$$

Τέλος, μπορώ να δω ότι εάν μετατοπίσω τον c_3 κατά 2, θα έχω το m .

$$m = c_3 \ll 2 \quad (10)$$

$$(10) \Rightarrow m = (m_0 \ m_1 \ m_2 \ m_3 \ m_4 \ \dots \ m_{11} \ m_{12} \ m_{13} \ m_{14} \ m_{15})$$

3.2.2 Λεπτομέρειες για την υλοποίηση

Εκτέλεση αρχείου:

python >> cyclic_shift.py

Ο κλειστός τύπος για την εύρεση του μηνύματος είναι

$$m = (((c \oplus (c \ll 10)) \oplus ((c \oplus (c \ll 10)) \ll 2)) \oplus (c \ll 14)) \ll 2$$

Έχουνε δημιουργηθεί τρεις συναρτήσεις. Μια που υλοποιεί την πράξη XOR, μια την πράξη Left - Shift και μια που παράγει τυχαία δυαδικά μηνύματα.

Αφού παραχθεί πρώτα ένα τυχαίο δυαδικό μήνυμα μήκους 16-bits, κρυπτογραφείται με την μέθοδο της εκφώνησης και στην συνέχεια αποκρυπτογραφείται με την μέθοδο που αναπτύχθηκε προηγουμένως.

3.2.3 Συμπέρασμα

Μπορούμε να αποφανθούμε, πως η μέθοδος αποκρυπτογράφησης αποτελεί ένα τρόπο για να βρούμε το αρχικό μήνυμα, καθώς ακολουθώντας διαφορετική συλλογιστική πορεία στις πράξεις μπορούμε και πάλι να βρούμε το αρχικό μήνυμα.

4 Τέλεια ασφάλεια

4.1 Εκφώνηση

Να αποδείξετε ότι, αν στο σύστημα μετατόπισης διαλέγουμε τυχαία τα κλειδιά από το σύνολο $\{0, 1, \dots, 23\}$, τότε το σύστημα έχει τέλεια ασφάλεια.

4.2 Λύση

4.2.1 Μαθηματική επίλυση

Για να έχει ένα σύστημα τέλεια ασφάλεια (Perfect Secrecy) θα πρέπει κατά C. Shannon να ισχύει το εξής (με την βοήθεια των σημειώσεων της διάλεξης [2]):

Ορισμός. Έστω $(\mathcal{M}, \mathcal{C}, \mathcal{K}, E, D)$ έχει τέλεια ασφάλεια, αν $\forall (m_1, m_2) \in \mathcal{M}$ και για $c \in \mathcal{C}$ ισχύει:

$$Pr[k \leftarrow \mathcal{K} : E(m_1, k) = c] = Pr[k \leftarrow \mathcal{K} : E(m_2, k) = c], \forall k \xleftarrow{\$} \mathcal{K} \quad (11)$$

Θα εξετασθεί για αρχή, τα μηνύματα μήκους ενός χαρακτήρα ($l = 1$).

Απόδειξη. Για κάθε γραμμα m και $c \in \mathcal{C}$ όπου $\mathcal{C} = \mathcal{M} = \{A, B, \dots, \Omega\}$, υπάρχει μοναδικό $k = c - m \pmod{24}$ τ.ω. $E(m, k) = m + k \pmod{24} = c$. Συνεπώς, για κάθε m, c ισχύει $Pr_{k \leftarrow \mathcal{K}}[k \leftarrow \mathcal{K} : E(m, k) = c] = 1/24$, όπου πληρεί τον ορισμό του C. Shannon. \square

Στην συνέχεια θα εξετασθεί για μηνύματα μήκους μεγαλύτερο από 1 ($l > 1$).

Απόδειξη. Έστω $m_1 = AB$, $m_2 = A\Omega$ και $c = B\Gamma$. Τότε υπάρχει κλειδί $k \in \mathcal{K}$ τ.ω. $E(m_1, k) = c$, για $k = 1$. Ωστόσο, για κάθε $k \in \mathcal{K}$ υπάρχει $E(m_2, k) \neq c$ και συνεπώς $Pr_{K \leftarrow \mathcal{K}}[E(m_1, K) = c] = 1/24$, όμως $Pr_{K \leftarrow \mathcal{K}}[E(m_2, K) = c] = 0$ άρα, δεν πληρείτε ο ορισμός C. Shannon. \square

4.2.2 Συμπέρασμα

Από τα παραπάνω, καταλήγουμε στο ότι ένα σύστημα μετατόπισης μπορεί να έχει τέλεια ασφάλεια αν-ν το μέγεθος το μηνύματος είναι ίσο με 1.

5 One Time Pad

5.1 Εκφώνηση

Υλοποιήστε τον OTP αφού αρχικά μετατρέψετε το μήνυμά σας σε bit με χρήση του παρακάτω πίνακα. Θα πρέπει να δουλεύει η κρυπτογράφηση και η αποκρυπτογράφηση. Το μήνυμα δίνεται κανονικά και έσωτερικά μετατρέπεται σε bits. Το κλειδί είναι διαλεγμένο τυχαία και έχει μήκος όσο το μήκος του μηνύματος σας. Το αποτέλεσμα δίνεται όχι σε bits αλλά σε λατινικούς χαρακτήρες.

5.2 Λύση

5.2.1 Λεπτομέρειες για την υλοποίηση

Εκτέλεση αρχείου:

```
python >> otp.py >> message
```

Τα argument <message> είναι προαιρετικό σε περίπτωση που θέλει ο χρήστης να χρησιμοποιήσει το One Time Pad με κάποιο δικό του μήνυμα.

5.2.2 Συμπέρασμα

Δίνοντας το μήνυμα:

MISTAKES ARE AS SERIOUS AS THE RESULTS THEY CAUSE

Για κάποια εκτέλεση του κώδικα (καθώς βασίζεται στην ψευδοτυχειότητα) έχουμε το κρυπτογραφημένο μήνυμα:

TR- !W! (UOYPLS)LRNLZSSF?LLXYY)HFSPTPCGOJVS

Όπου θα αποκρυπτογραφηθεί ως εξής:

MISTAKESAREASSERIOUSASTHERESULTSTHEYCAUSE

6 Η αριθμοθεωρητική συνάρτηση Möbius

6.1 Εκφώνηση

Αποδεικνύεται ότι το πλήθος των ανάγωγων πολυωνύμων βαθμού n στο σώμα \mathbb{F}_2 είναι

$$N_2(n) = \frac{1}{n} \sum_{d|n} \mu(d) \cdot 2^{n/d}, \quad (12)$$

όπου

$$\mu(d) = \begin{cases} 1 & d = 1 \\ (-1)^k & d = p_1 p_2 \cdots p_k \text{ πρώτοι} \\ 0 & \text{αλλού} \end{cases} \quad (13)$$

Με το σύμβολο $d|n$ εννοούμε όλους τους θετικούς διαιρέτες του n . π.χ. αν $n = 30$, τότε

$$\{d|n : 1 \leq d \leq n\} = \{1, 2, 3, 5, 6, 10, 15, 30\} \quad (14)$$

Με χρήση του συστήματος sagemath υπολογίστε το $N_2(10)$.

6.2 Λύση

6.2.1 Λεπτομέρειες για την υλοποίηση

```
python >> moebius.py >> n
```

Το argument `<n>` είναι προαιρετικό σε περίπτωση που θέλει ο χρήστης να υπολογίσει την σχέση 12 με άλλο n , οι προκαθορισμένη τιμή είναι αυτή της εκφώνησης.

Δημιουργήθηκαν τρεις βασικές συναρτήσεις, μια για τον υπολογισμό του $\mu(d)$, μια για τον υπολογισμό του συνόλου $d|n$ και μια για την τιμή $N_2(n)$.

6.2.2 Συμπέρασμα

Η τιμή που υπολογίσθηκε είναι $N_2(10) = 99$.

7 Rivest Cipher 4 (RC4)

7.1 Εκφώνηση

Υλοποιήστε τον RC4. Χρησιμοποιώντας το κλειδί HOUSE κρυπτογραφήστε το μήνυμα (ξαναγράψτε το χωρίς κενά):

MISTAKES ARE AS SERIOUS AS THE RESULTS THEY CAUSE

Η υλοποίηση σας πρέπει και να αποκρυπτογραφεί σωστά.

7.2 Λύση

7.2.1 Λεπτομέρειες για την υλοποίηση

Εκτέλεση αρχείου:

```
python python rc4.py message key
```

Τα arguments <message> και <key> είναι προαιρετικά σε περίπτωση που θέλει ο χρήστης να το RC4 με άλλες τιμές, οι προκαθορισμένες τιμές είναι αυτές που δίνονται στην εκφώνηση.

7.2.2 Συμπέρασμα

Εκτελώντας το αρχείο για τις τιμές της εκφώνησης, επιστρέφεται το εξής κρυπτογραφημένο μήνυμα:

IGD!APO-TJUQPDSPMAOZUIAZ(VF(VFGQ.IIWMB(WX

Ενώ η αποκρυπτογραφημένη μορφή του παραπάνω μηνύματος είναι:

MISTAKESAREASSERIOUSASTHERESULTSTHEYCAUSE

8 Διαφορική ομοιομορφία

8.1 Εκφώνηση

Αν Σ ένα σύνολο με $|\Sigma|$ συμβολίζουμε το πλήθος των στοιχείων του, υπολογίστε τη διαφορική ομοιομορφία (differential uniformity) του S-box,

$$Diff(S) = \max_{x \in \{0,1\}^6 - \{0\}, y \in \{0,1\}^4} |\{z \in \{0,1\}^6 : S(z \oplus x) \oplus S(z) = y\}| \quad (15)$$

Γενικά, για S-boxes:

$$S : \{0,1\}^n \rightarrow \{0,1\}^m \quad (16)$$

ο προηγούμενος ορισμός γράφεται:

$$Diff(S) = \max_{x \in \{0,1\}^n - \{0\}, y \in \{0,1\}^m} |\{z \in \{0,1\}^n : S(z \oplus x) \oplus S(z) = y\}| \quad (17)$$

και ισχύει:

$$Diff(S) \geq \max\{2, 2^{n-m}\} \quad (18)$$

Όσο μικρότερη είναι αυτή η ποσότητα, τόσο πιο ανθεκτικό είναι το S-box στη διαφορική κρυπτανάλυση.

8.2 Λύση

8.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- matplotlib
- matplotlib.pyplot

Εκτέλεση αρχείου:

python >> sbox.py

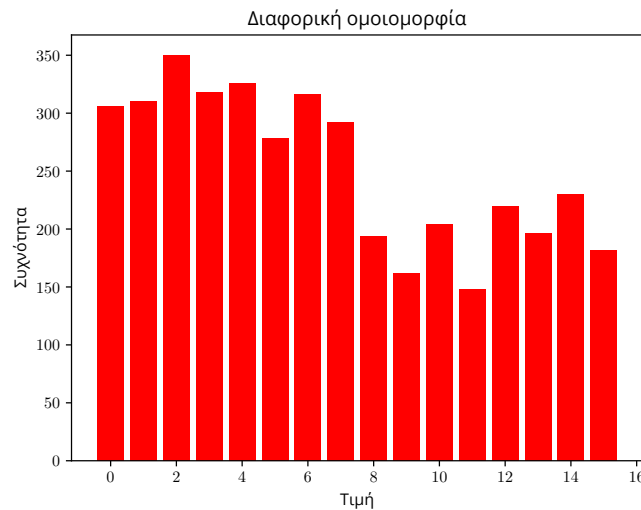
Το αρχείο, περιέχει ουσιαστικά την υλοποίηση της σχέσης (15). Η υλοποίηση έγινε με βάση το βιβλίο [3] που χρησιμοποιηθεί την συνάρτηση Feistel που περιγράφεται μέσα.

8.2.2 Συμπέρασμα

Για το S-box της εκφώνησης ισχύει: $n = 6, m = 4$. Έτσι η σχέση (18) έχει ως εξής:

$$Diff(S) \geq \max\{2, 2^{n-m}\} = \max\{2, 2^{6-4}\} = \max\{2, 4\} = 4 \Rightarrow Diff(S) \geq 4 \quad (19)$$

Δηλαδή, αναμένεται η διαφορική ομοιομορφία να έχει τιμή τουλάχιστον ίση με 4. Εκτελώντας τον κώδικα, παράγεται το εξής γράφημα:



Σχήμα 1: Διαφορική ομοιομορφία του S-box (15)

Όπως φαίνεται λοιπόν από το σχήμα 1 η μέγιστη συχνότητα εμφάνισης είναι το 350 για όλες τις πιθανές τιμές που μπορεί να πάρει το x (πλήθος τιμών του x είναι $2^6 - 1 = 63$), δηλαδή $Diff(S) = \max\{|S(z \oplus x) \oplus S(z)|\} = 350$.

9 Avalanche effect

9.1 Εκφώνηση

Εξετάστε αν ισχύει το avalanche effect στο AES-128. Αναλυτικότερα, φτιάξτε αρκετά ζευγάρια (≥ 30) μηνυμάτων (m_1, m_2) που να διαφέρουν σ ένα bit. Εξετάστε σε πόσα bits διαφέρουν τα αντίστοιχα κρυπτομηνύματα. Δοκιμάστε με δύο καταστάσεις λειτουργίας: ECB και CBC (η δεύτερη θέλει και IV block). Τα μήκη των μηνυμάτων που θα χρησιμοποιήσετε να έχουν μήκος διπλάσιο του μήκους ενός block. Δηλαδή για τον AES-128, να είναι μήκους 256-bits.

9.2 Λύση

9.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- Crypto.Cipher
- bitstring
- statistics

Εκτέλεση αρχείου:

```
python avalanche.py
```

Δοκιμάσθηκαν 200 τυχαία ζευγάρια δυαδικών μηνυμάτων, χρησιμοποιώντας τυχαία δυαδικών κλειδιά μήκους 128-bit αντίστοιχα και για το IV. Για την χρήση του AES, έγινε χρήση της βιβλιοθήκης **Crypto** όπου εξετάσθηκαν και οι δυο λειτουργίες.

9.2.2 Συμπέρασμα

Έπειτα από αρκετές προσομοιώσεις, οι μέσοι όροι διαφορών στα ζεύγη των μηνυμάτων έχουν ως εξής: για ECB 64-bit και για CBC 128-bit. Αυτό σημαίνει πως στην λειτουργία ECB η αλλαγή ενός bit επιφέρει αλλαγή στο $\approx 25\%$ του μηνύματος άρα, δεν παρουσιάζεται το avalanche effect σε αυτή την λειτουργία. Αντιθέτως, στην λειτουργία CBC αλλάζει το $\approx 50\%$ του μηνύματος που σημαίνει ότι υπάρχει το avalanche effect.

10 Capture The Flag

10.1 Εκφώνηση

Μπορείτε να ανοίξετε το secure.zip?

10.2 Λύση

```
be121740bf988b2225a313fa1f107ca1
```

10.3 Πορεία

- aHR0cHM6Ly9jcmlwdG9sb2d5LmNzZC5hdXRoLmdyOjgwODAvG9tZS9wdWIvMTUv
- <https://cryptology.csd.auth.gr:8080/home/pub/15/>
- #heretic!
- <https://tinyurl.com/26ru4359>
- f1f5e44313a1b684f1f7e8eddec4fcb0

Μέρος II

Κρυπτογραφία Δημοσίου Κλειδιού

11 Diffie-Hellman

11.1 Εκφώνηση

Εφαρμόστε το πρωτόκολλο ανταλλαγής Diffie-Hellman για $g = 3$, $p = 101$, $a = 77$, $b = 91$. Δηλ., πρέπει να υπολογίσετε το κοινό κλειδί. Θα χρειαστεί να υλοποιηθεί Αλγόριθμος (7.2.2) από [3, σελ. 82].

11.2 Λύση

11.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- frow (ορισμένη από τον συντάκτη)

Εκτέλεση αρχείου:

```
python >> diffie_hellman.py
```

Αποτελεί μια απλή εφαρμογή του πρωτοκόλλου Diffie-Hellman, κάνοντας χρήση της γρήγορης ύψωσης σε δύναμη που θα δούμε και στην επόμενη άσκηση.

11.2.2 Συμπέρασμα

Το κοινό κλειδί είναι 66.

12 Γρήγορη ύψωση σε δύναμή

12.1 Εκφώνηση

Υλοποιήστε σε όποια γλώσσα προγραμματισμού θέλετε τον Αλγόριθμο (7.2.2) από [3, σελ. 82] και κατόπιν υπολογίστε τη δύναμη $5^{77} \pmod{19}$.

12.2 Λύση

12.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- math

Εκτέλεση αρχείου:

python > fpow.py

Ακολουθήθηκε η υλοποίηση ένα προς ένα, από [3] στην σελίδα 82.

12.2.2 Συμπέρασμα

Η εκτέλεση της γρήγορης ύψωση σε δύναμη για τις δεδομένες τιμές επιστρέφει την τιμή 9, που εύκολα επαληθεύεται ως ορθή.

13 Πρώτοι αριθμοί

13.1 Εκφώνηση

Να αποδείξετε ότι ο n -οστος πρώτος που ικανοποιεί την ανισότητα

$$p_n < 2^{2^n} \quad (20)$$

13.2 Λύση

13.2.1 Μαθηματική επίλυση

Υποθέτουμε ότι $p_i \leq 2^{2^i} \forall i \leq n$. Στην συνέχεια, το $p_1 p_2 \dots p_n + 1$ δεν διαιρείται από κανένα από τους πρώτους αριθμούς p_1, \dots, p_n , έτσι

$$p_{n+1} \leq \prod_{i=1}^n p_i + 1 \quad (21)$$

Αντικαθιστώντας τις προηγούμενες ανισώσεις για το p_i και χρησιμοποιώντας $\sum_{i=1}^n 2^i = 2^{n+1} - 2$, έχουμε

$$p_{n+1} \leq \left(\prod_{i=1}^n 2^{2^i} \right) + 1 = (2^{\sum_{i=1}^n 2^i}) + 1 = 2^{2^{n+1}-2} + 1 \leq 2^{2^{n+1}} \quad (22)$$

14 Ιδιότητες του gcd

14.1 Εκφώνηση

Έστω $a, b \in \mathbb{Z}^+$. Αν $\gcd(a, b) = 1$, τότε

- $\forall c \in \mathbb{Z}$ ισχύει $\gcd(ac, b) = \gcd(c, b)$
- $\gcd(a+b, a-b) \in \{1, 2\}$. Ειδικότερα, ν.α.ο αν a, b περιττοί θετικοί ακέραιοι $\gcd(a+b, a-b) = 2$

- $\gcd(2^a - 1, 2^b - 1) = 1$
- $\gcd(M_p, M_q) = 1$ όπου $M_p = 2^p - 1, M_q = 2^q - 1$ Mersenne ακέραιοι (p, q πρώτοι με $p \neq q$).

14.2 Λύση

14.2.1 Μαθηματική επίλυση

Πρόταση. $\forall c \in \mathbb{Z}$ ισχύει $\gcd(ac, b) = \gcd(c, b)$

Απόδειξη. Έστω ότι $d_1 = \gcd(c, b)$ και $d_2 = \gcd(ac, b)$. Τότε έχουμε $cx_1 + by_1 = d_1, acx_2 + by_2 = d_2$ και $ax + by = 1$ από Bezout. Αρχικά πολλαπλασιάζουμε την $ax + by = 1$ με το d_1 για να δείξουμε $d_2 | d_1$.

$$\begin{aligned} d_1(ax + by) &= 1 \\ \Rightarrow ax(cx_1 + by_1) + bd_1y &= d_1 \\ \Rightarrow ac(xx_1) + b(axy_1 + d_1y) &= d_1 \end{aligned} \quad (23)$$

Εφόσον ισχύει ότι $d_2 = \gcd(ac, b)$ τότε διαιρεί κάθε ακέραιο γραμμικό συνδυασμό των ac και b και άρα έχουμε $d_2 | d_1$.

Με παρόμοιο τρόπο, θα πολλαπλασιάσουμε το $ax + by = 1$ με το d_2 και έχουμε

$$\begin{aligned} d_2(ax + by) &= 1 \\ \Rightarrow ax(acx_2 + by_2) + bd_2y &= d_2 \\ \Rightarrow c(a^2xx_2) + b(axy_2 + d_2y) &= d_2 \end{aligned} \quad (24)$$

Ομοίως με πριν ισχύει ότι $d_1 = \gcd(c, b)$, τότε διαιρεί κάθε ακέραιο γραμμικό συνδυασμό των ac και b άρα έχουμε $d_1 | d_2$.

Τελικά, επειδή έχουμε $d_1 | d_2, d_2 | d_1$ και d_1 και d_2 είναι μη-αρνητικοί (αφού είναι το gcd δύο ακεραίων), καταλήγουμε ότι $d_1 = d_2$.

Επομένως, $\gcd(ac, b) = \gcd(c, b)$. □

Πρόταση. $\gcd(a + b, a - b) \in \{1, 2\}$

Απόδειξη. Έστω d κοινός διαιρέτης των $a + b$ και $a - b$, τότε ο d διαιρεί και το άθροισμα και την διαφορά τους

$$\begin{aligned} d &| (a + b) \\ d &| (a - b) \\ d &| (a + b) + (a - b) = 2a \\ d &| (a + b) - (a - b) = 2b \end{aligned}$$

Άρα, έχουμε

$$d | \gcd(2a + 2b) = 2 \gcd(a, b) \xrightarrow{1} d | 2 \quad (25)$$

που σημαίνει ότι το $d \in \{1, 2\}$. □

Πρόταση. Ν.α.ο αν a, b περιττοί θετικοί ακέραιοι $\gcd(a + b, a - b) = 2$

Απόδειξη. Έχουμε a, b περιττούς, που γράφονται ως εξής

$$a = 2k_1 + 1, k_1 \in \mathbb{Z}$$

$$b = 2k_2 + 1, k_2 \in \mathbb{Z}$$

Επίσης, είναι γνωστό πως το άθροισμα και η διαφορά δυο περιττών αριθμών είναι άρτιος αριθμός.

Αφού πρόκειται για δυο άρτιος αριθμούς, τότε και οι διαιρέτες τους θα είναι άρτιοι.

Παραπάνω, αποδείξαμε πως αν d διαιρέτης, τότε $d \in \{1, 2\}$ όπου μόνο για $d = 2$ άρτιος. \square

Πρόταση. $\gcd(2^a - 1, 2^b - 1) = 1$

Απόδειξη. Έστω $p = \gcd(a, b)$ τότε $p = ax + by$ με $x, y \in \mathbb{Z}$.

Αν, $d = \gcd(2^a - 1, 2^b - 1)$ τότε $2^a \equiv 1 \pmod{d}$ και $2^b \equiv 1 \pmod{d}$ έτσι έχουμε

$$2^p = 2^{ax+by} = (2^a)^x (2^b)^y \equiv 1 \pmod{d} \quad (26)$$

Συνεπώς $d | 2^p - 1$. Από την άλλη, αν $p | a$ τότε $2^p - 1 | 2^a - 1$ άρα $2^p - 1$ κοινός παράγοντάς.

Τελικά, $\gcd(2^a - 1, 2^b - 1) = 2^{\gcd(a,b)} - 1 \Rightarrow \gcd(2^a - 1, 2^b - 1) = 1$ \square

Πρόταση. $\gcd(M_p, M_q) = 1$ όπου $M_p = 2^p - 1, M_q = 2^q - 1$

Απόδειξη. Παρατηρείται ότι εάν αντικαταστήσω στην προηγούμενη απόδειξη με M_p, M_q ισχύει η παραπάνω σχέση. \square

15 Ακόμα ένα πρόβλημα με gcd

15.1 Εκφώνηση

Έστω a, b, c ακέραιοι και $\delta = a^2 - 4bc^2 \neq 0$. Ν.α.ο. $\gcd(\delta, 4c^2)$ είναι τετράγωνο.

15.2 Λύση

15.2.1 Μαθηματική επίλυση

Η λύση είναι αυτούσια από [4] του συναδέλφου κατά την διάρκεια εκπόνησης τούτης εργασίας.

Απόδειξη. Ισχύει γενικά πως

$$\gcd(a^n, b^n) = [\gcd(a, b)]^n, \forall a, b \in \mathbb{Z}$$

Αυτό φαίνεται από τις πρωτογενείς αναλύσεις των a, b :

$$a = p_1^{e_{11}} p_2^{e_{12}} p_3^{e_{13}}$$

$$b = p_1^{e_{21}} p_2^{e_{22}} p_3^{e_{23}}$$

$$\Rightarrow \gcd(a, b) = p_1^{\min(e_{11}, e_{21})} p_2^{\min(e_{12}, e_{22})} p_3^{\min(e_{13}, e_{23})} \quad (27)$$

Υψώνοντας σε κοινή δύναμη έχουμε:

$$a^n = (p_1^{e_{11}} p_2^{e_{12}} p_3^{e_{13}} \dots)^n = p_1^{ne_{11}} p_2^{ne_{12}} p_3^{ne_{13}} \dots$$

$$b^n = (p_1^{e_{11}} p_2^{e_{12}} p_3^{e_{13}} \dots)^n = p_1^{ne_{11}} p_2^{ne_{12}} p_3^{ne_{13}} \dots$$

$$\begin{aligned} \gcd(a^n, b^n) &= \\ p_1^{\min(ne_{11}, ne_{21})} p_2^{\min(ne_{12}, ne_{22})} p_3^{\min(ne_{13}, ne_{23})} &= \\ p_1^{n \min(e_{11}, e_{21})} p_2^{n \min(e_{12}, e_{22})} p_3^{n \min(e_{13}, e_{23})} &= \\ (p_1^{\min(e_{11}, e_{21})} p_2^{\min(e_{12}, e_{22})} p_3^{\min(e_{13}, e_{23})})^n \end{aligned} \quad (28)$$

Άρα, $\gcd(a^n, b^n) = [\gcd(a, b)]^n$.
Είναι επίσης τετριμμένο πως

$$\gcd(a + mb, b) = \gcd(a, b), \forall a, b, m \in \mathbb{Z}$$

Γνωρίζοντας τα παραπάνω, έχουμε

$$\gcd(a, 2c) = d \gcd(a^2, 4c^2) = d^2 \wedge d^2 | a^2 4bc^2 \Rightarrow \gcd(\delta, 4c^2) = d^2$$

□

16 Ιδιότητα των περιττών ακέραιων

16.1 Εκφώνηση

Επαληθεύστε πειραματικά ότι για όλους τους περιττούς ακέραιους $< 2^{20}$ ισχύει

$$\frac{\sigma(n)}{n} < \frac{e^\gamma}{2} \ln \ln n + \frac{0.74}{\ln \ln n} \quad (29)$$

16.2 Λύση

16.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- collections
- concurrent.futures
- trial_division (ορισμένη από τον συντάκτη)
- os
- math

Εκτέλεση αρχείου:

```
python >> sum.py
```

Για τον υπολογισμό του αθροίσματος των θετικών διαιρετών του κάθε αριθμού n , έγινε χρήση της δοκιμαστικής διαίρεσης ε.ω. να βρεθούνε για κάθε n οι διαιρέτες του.

Στην συνέχεια υπολογίστηκε ο τύπος

$$\sigma(n) = \prod_{i=1}^k \frac{p_i^{a_i+1} - 1}{p_i - 1}$$

όπου p_i οι πρώτοι διαιρέτες και a_i η μεγαλύτερη δύναμη του p_i .

Για να εξετασθούν οι περιττοί αριθμοί γρήγορα, έγινε υλοποίηση που εκτελείτε σε παραλληλία χρησιμοποιώντας τους ελεύθερους διαθέσιμους πυρήνες του επεξεργαστή.

16.2.2 Συμπέρασμα

Εκτελώντας το πρόγραμμα καταλήγουμε στο ότι η υπόθεση της εκφώνησης ισχύει.

17 Carmichael

17.1 Εκφώνηση

N.α.ο οι αριθμοί 9999109081, 6553130926752006031481761 είναι αριθμοί Carmichael. Μπορείτε να βρείτε κάποιον μεγαλύτερο; Υποδ. Να γίνει χρήση του κριτηρίου Korselt.

17.2 Λύση

17.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- `sympy.ntheory`
- `fermat` (ορισμένη από τον συντάκτη)

Εκτέλεση αρχείου:

```
python >> carmichael.py
```

Μιας και δεν είναι ο στόχος της άσκησης η παραγοντοποίηση ενός αριθμού, χρησιμοποιήθηκε για λόγους επιτάχυνσης εκτέλεσης του προγράμματος η μέθοδος `factorint` της `sympy.ntheory`. Αφού βρεθούν οι παράγοντες του αριθμού, εξετάζεται μέσω συναρτήσεων εάν υπάρχει κάποιος παράγοντας που εμφανίζεται περισσότερες από μια φορές, εάν όλοι οι παράγοντες μειωμένοι κατά την μονάδα διαιρούν τον αριθμό μειωμένο κατά την μονάδα και τέλος εάν ο αριθμός αποτελεί σύνθετο αριθμό.

Για την διαπίστωση εάν ο αριθμός είναι σύνθετος, χρησιμοποιήθηκε το τεστ του Fermat, οποίος επιστρέφει αποτελέσματα όχι με βεβαιότητά αλλά με κάποια πιθανότητα καθώς βασίζεται στη τυχαιότητα. Αυτό σημαίνει, πως για να αποφανθούμε στο εάν οι αριθμοί είναι σύνθετοι ή πρώτοι έπρεπε να εκτελεσθεί το τεστ αρκετές φορές.

17.2.2 Συμπέρασμα

Και οι δυο δοσμένοι αριθμοί, πληρούν τις συνθήκες του Korselt άρα είναι αριθμοί Carmichael.

18 Τεστ του Fermat

18.1 Εκφώνηση

Ικανοποιούν το τεστ ου Fermat οι αριθμοί

$$835335 \cdot 2^{39014} \pm 1;$$

18.2 Λύση

Βιβλιοθήκες

- gmpy2

Εκτέλεση αρχείου:

```
python >> fermat.py
```

Η υλοποίηση ακολουθεί τον αλγόριθμο που δίνεται από το βιβλίο [3, σελ. 122], προσθέτοντας τον έλεγχο για το εάν ο αριθμός είναι άρτιος δεν μπορεί να είναι και πρώτος.

18.2.1 Συμπέρασμα

Εκτελώντας τον κώδικα, επιστρέφεται ως και οι δυο αριθμοί είναι πρώτοι αριθμοί.

19 Trial Division

19.1 Εκφώνηση

Να υλοποιήσετε τον αλγόριθμο δοκιμαστικής διαίρεσης και να παραγοντοποιήσετε τους αριθμούς $2^{62} - 1$, $2^{102} - 1$.

19.2 Λύση

19.2.1 Λεπτομέρειες για την υλοποίηση

Εκτέλεση αρχείου:

```
python >> trial_division.py
```

Ακολουθήθηκε πιστά ο αλγόριθμος δόθηκε στο βιβλίο [3] στην σελίδα 134.

19.2.2 Συμπέρασμα

Οι αριθμοί παραγοντοποιούνται ως εξής

$$2^{62} - 1 = \{3, 3, 3, 3, 7, 19, 73, 87211, 67240192\}$$

και για

$$2^{102} - 1 = \{3, 3, 3, 3, 7, 19, 73, 87211, 262657, 281474976710656\}$$

20 Αλγόριθμος του Lehman

20.1 Εκφώνηση

Να υλοποιήσετε τον αλγόριθμο του Lehman. Κατόπιν, διαλέξτε ένα τυχαίο ακέραιο με 100 bits. Θεωρούμε επιτυχία, αν ο αλγόριθμος σας παραγοντοποιήσει τον τυχαίο ακέραιο σε λιγότερο από 10 δευτερόλεπτα στον Η/Υ. Εκτελέσετε το προηγούμενο πείραμα 100 φορές. Ποιο το ποσοστό επιτυχίας;

20.2 Λύση

Βιβλιοθήκες

- gmpy2
- sympy
- multiprocessing

Εκτέλεση αρχείου:

```
python > lehman.py
```

Υλοποιήθηκε ο αλγόριθμος όπως παρουσιάζεται στο [3, σελ. 136]. Για να επιτευχθεί υψηλότερη ταχύτητα, μέσω συνάρτησης της `sympy`, επιστρέφονται όλοι οι πρώτοι μέχρι ένα σημείο για να εξετασθούν από την δοκιμαστική διαίρεση για το εάν είναι διαιρέτης του αριθμού. Επιπλέον, για να εκτελεσθούν οι 1000 δοκιμές σε πλαίσιο 10 δευτερολέπτων χρησιμοποιήθηκαν νήματα.

20.2.1 Συμπέρασμα

Επειδή βαδίζομαστε στην τυχαιότητα στην διαλογή των αριθμών τα αποτελέσματα ποικίλουν. Ωστόσο, καταλήγουμε σε ένα αποτέλεσμα επιτυχίας $\approx 90\%$.

21 Αλγόριθμος του Pollard

21.1 Εκφώνηση

Σε αυτή την άσκηση θα δούμε πως μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο του Pollard για παραγοντοποίηση. Θεωρούμε ένα φυσικό N και ζητάμε ένα διαιρέτη του. Έστω $f(x) = (x^2 + 1) \bmod N$ και ας είναι το x_0 τυχαία τιμή από το $\{2, 3, \dots, N - 1\}$. Η γραμμή 6 στον Αλγόριθμο (10.2.2) θα είναι

$$1 < \gcd(|x - y|, N) < N.$$

Υλοποιήστε τον αλγόριθμο και βρείτε ένα διαιρέτη του αριθμού $N = 2^{257} - 1$.

21.2 Λύση

21.2.1 Λεπτομέρειες για την υλοποίηση

Εκτέλεση αρχείου:

```
python > pollard.py
```

Ακολουθήθηκε πιστά ο αλγόριθμος δόθηκε στο βιβλίο [3, σελ. 150], προσθέτοντας την συνθήκη της εκφώνησης.

21.2.2 Συμπέρασμα

Για $N = 2^{257} - 1$, ο αλγόριθμος του Pollard επιστρέφει 535006138814359.

22 RSA

22.1 Εκφώνηση

Δίνεται το δημόσιο κλειδί $(N, e) = (11413, 19)$. Βρείτε το ιδιωτικό κλειδί και κατόπιν αποκρυπτογραφήστε το μήνυμα

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

Υποθέστε ότι τα γράμματα στο αρχικό μήνυμα m , αναπαρίστανται από τις ASCII τιμές τους (δουλέψτε block by block το C). Υποδ. Παραγοντοποιήστε N , κατόπιν υπολογίστε το $\phi(N)$.

22.2 Λύση

22.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- gmpy2
- numpy
- frow

Εκτέλεση αρχείου:

```
python>>rsa.py
```

Για να προχωρήσουμε σε αποκρυπτογράφηση του C , θα πρέπει για κάθε block του μηνύματος να λυθεί η εξίσωση

$$m_i = c_i^d \mod N, \forall c_i \in C \quad (30)$$

όπου το N ήδη γνωστό ενώ, το d θα πρέπει να βρεθεί. Για την εύρεση του d , θα πρέπει να γίνει παραγοντοποίηση του N χρησιμοποιώντας την μέθοδο του Fermat [5] δηλ. να βρεθούν p, q ε.ω. $N = p \cdot q$ και στην συνέχεια να υπολογισθεί η $\phi(N) = (p-1) \cdot (q-1)$. Τέλος, θα πρέπει να λυθεί η εξίσωση

$$e \cdot d \equiv 1 \mod \phi(N) \Rightarrow d = e^{-1} \mod \phi(N) \quad (31)$$

Υπολογίζοντας την παραπάνω σχέση μέσω κατάλληλων συναρτήσεων, προκύπτει ότι $d = 1179$.

22.2.2 Συμπέρασμα

Εκτελώντας την σχέση (30) για κάθε block, επιστρέφεται το μήνυμα

welcove to the real world

23 Wiener Attack

23.1 Εκφώνηση

Ας θεωρήσουμε $(N, e) = (194749497518847283, 50736902528669041)$ και το κρυπτογραφημένο κείμενο C , που έχει προκύψει από το Textbook RSA και έπειτα κωδικοποιήθηκε. Εφαρμοστέ την επίθεση Wiener, για να βρείτε το κλειδί d . Υποθέτουμε ότι στο αρχικό κείμενο m κάθε χαρακτήρας έχει αντικατασταθεί από την ASCII τιμή του. Τέλος, βρείτε το αρχικό m .

23.2 Λύση

23.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- gmpy2
- base64
- re

Εκτέλεση αρχείου:

```
python wiener.py
```

Η επίθεση του Wiener, επιδιώκει να βρει το d κάνοντας χρήση συνεχών κλασμάτων. Αρχικά, υπολογίζω για αρχή το ανάπτυγμα σε συνεχές κλάσμα του $\frac{e}{N}$. Στην συνέχεια, βρίσκω όλα τα ανάγωγα κλάσματα $\frac{N_i}{D_i}$ που λέγονται συγκλίνοντα κλάσματα. Στην συνέχεια υπολογίζονται ϕ_i, p, q κάνοντας χρήση των ανάγωγων κλασμάτων και τέλος η εξίσωση $x^2 - ((N_i - \phi_i + 1)x + N_i) = 0$. Ακολουθώντας την παραπάνω διαδικασία, ευρίσκεται ότι $d = 20881$.

Για να προχωρήσουμε στην αποκρυπτογράφηση του μηνύματος, θα πρέπει πρώτα να αποκωδικοποιηθεί με την χρήση της μεθόδου Base64.

23.2.2 Συμπέρασμα

Εκτελώντας λοιπόν την επίθεσή του Wiener, το αρχικό μήνυμα είναι

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

24 Naive RSA

24.1 Εκφώνηση

Δίνεται ότι δημόσιο κλειδί μιας (naive) ψηφιακής υπογραφής RSA, $(N, e) = (9899, 839)$. Αν η ψηφιακή υπογραφή του μηνύματος $m = 3$ είναι $s = 301$, μπορείτε να επαληθεύσετε ότι είναι σωστή η ψηφιακή υπογραφή;

24.2 Λύση

24.2.1 Λεπτομέρειες για την υλοποίηση

Βιβλιοθήκες

- fpow

Εκτέλεση αρχείου:

```
python >> signature.py
```

Για να γίνει επαλήθευση της υπογραφής θα πρέπει να ισχύει

$$a = m, \text{ όπου } a = s^e \pmod{N}. \quad (32)$$

24.2.2 Συμπέρασμα

Για τις τιμές της εκφώνησης, δεν μπορεί να επαληθευτεί η υπογραφή.

25 gpg

25.1 Εκφώνηση

Στείλτε ένα κρυπτογραφημένο μήνυμα στον συγγραφέα (το δημόσιο κλειδί που έχει αναγνωριστικό `0xEB1185F82713D6DF`). Από το δημόσιο κλειδί το αναγνωριστικό προκύπτει (σε bash) αν δώσουμε τον παρακάτω κώδικα

```
$gpg --list-packets pk.asc | awk '/keyid:/{ print $2 }'
```

Κώδικας 1: gpg κώδικας σε bash

25.2 Λύση

25.2.1 Συμπέρασμα

```
d41d8cd98f00b204e9800998ecf8427e
```

Αναφορές

- [1] T. M. Doctor, "Five ways to crack a vigenère cipher," 2020. <https://www.cipherchallenge.org/wp-content/uploads/2020/12/Five-ways-to-crack-a-Vigenere-cipher.pdf>.
- [2] S. Jarecki, "Crypto overview, perfect secrecy, one-time pad." Cryptology Course, Lecture Slides, 2004. <https://www.ics.uci.edu/%7Estasio/fall04/lect1.pdf>.
- [3] Κ. Δραζιώτης, *Εισαγωγή στην Κρυπτογραφία*. Κάλλιπος, Ανοικτές Ακαδημαϊκές Εκδόσεις, 2022. <https://dx.doi.org/10.57713/kallipos-17>.
- [4] Α. Κύδρος, "Κρυπτογραφία Δημοσίου Κλειδιού." Cryptology Course Assignment, 2023. https://github.com/asimakiskydros/University-Projects/blob/main/Cryptography/project%20%232/latex/project_2_3881.pdf.
- [5] H. Böck, "Fermat factorization in the wild." Cryptology ePrint Archive, Paper 2023/026, 2023. <https://eprint.iacr.org/2023/026>.