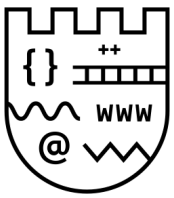

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Πληροφορικής

# Τεχνική αναφορά για NGE-06-03

*Ασύμμετρη Κρυπτογραφία*

Αλέξανδρος Κόρκος
**alexkork@csd.auth.gr**
**3870**

Θεσσαλονίκη, 11 Ιουνίου 2023

# Περιεχόμενα

# 1 Diffie-Hellman

## 1.1 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 [1] στην σελίδα 82).

## 1.2 Λύση

### 1.2.1 Λεπτομέρειες για την υλοποίηση

**Βιβλιοθήκες**

- fpow (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.

### 1.2.2 Συμπέρασμα

Το κοινό κλειδί είναι $66$.

# 2 Γρήγορη ύψωση σε δύναμή

## 2.1 Εκφώνηση

Υλοποιήστε σε όποια γλώσσα προγραμματισμού θέλετε τον Αλγόριθμο (7.2.2) (Από [1] στην σελίδα 82) και κατόπιν υπολογίστε τη δύναμη $5^{77} \pmod{19}$.

## 2.2 Λύση

### 2.2.1 Λεπτομέρειες για την υλοποίηση

**Βιβλιοθήκες**

- math

Εκτέλεση αρχείου:

```
python fpow.py
```

Ακολουθήθηκε η υλοποίηση ένα προς ένα, από [1] στην σελίδα 82.

#### 2.2.2 Συμπέρασμα

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

# 3 Πρώτοι αριθμοί

## 3.1 Εκφώνηση

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

$$p_n < 2^{2^n} \tag{1}$$

## 3.2 Λύση

### 3.2.1 Μαθηματική επίλυση

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

$$p_{n+1} \leq \prod_{i=1}^{n} p_i + 1 \tag{2}$$

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

$$p_{n+1} \leq (\prod_{i=1}^{n} 2^{2^i}) + 1 = (2^{\sum_{i=1}^{n} 2^i}) + 1 = 2^{2^{n+1}-2} + 1 \leq 2^{2^{n+1}} \tag{3}$$

# 4 Ιδιότητες του gcd

## 4.1 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 $p \neq q$).

## 4.2 Λύση

### 4.2.1 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 $d_2 | d_1$.

$$\begin{aligned} & d_1(ax + by = 1) \\ \Rightarrow\ & ax(cx_1 + by_1) + bd_1y = d_1 \\ \Rightarrow\ & ac(xx_1) + b(axy_1 + d_1y) = d_1 \end{aligned} \tag{4}$$

Εφόσον ισχύει ότι $d_2 = \gcd(ac, b)$ τότε διαίρει 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

$$\begin{aligned} & d_2(ax + by = 1) \\ \Rightarrow\ & ax(acx_2 + by_2) + bd_2y = d_2 \\ \Rightarrow\ & c(a^2xx_2) + b(axy_2 + d_2y) = d_2 \end{aligned} \tag{5}$$

Ομοίως με πριν ισχύει ότι $d_1 = \gcd(c, b)$, τότε διαίρει 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, ο $\gcd(ac, b) = \gcd(c, b)$. □

**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

$$d | (a + b)$$

$$d | (a - b)$$

$$d | (a + b) + (a - b) = 2a$$

$$d | (a + b) - (a - b) = 2b$$

Άρα, έχουμε

$$d | \gcd(2a + 2b) = 2\cancelto{1}{\gcd(a, b)} \Rightarrow d | 2 \tag{6}$$

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. □

**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

$$2^p = 2^{ax+by} = (2^a)^x (2^b)^y \equiv 1 \pmod d \tag{7}$$

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

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

*Απόδειξη.* Παρατηρείται ότι εάν αντικαταστήσω στην προηγουμένη απόδειξή με $M_p, M_q$ ισχύει η παραπάνω σχέση. □

# 5 Ακόμα ένα πρόβλημα με gcd

## 5.1 Εκφώνηση

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

## 5.2 Λύση

# 6 Ιδιότητα των περιττών ακέραιων

## 6.1 Εκφώνηση

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

$$\frac{\sigma(n)}{n} < \frac{e^\gamma}{2} \ln \ln n + \frac{0.74}{\ln \ln n} \tag{8}$$

## 6.2 Λύση

### 6.2.1 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.

### 6.2.2 Συμπέρασμα

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

# 7 Carmichael

## 7.1 Εκφώνηση

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

## 7.2 Λύση

### 7.2.1 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 φορά, 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 άλλα 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.

### 7.2.2 Συμπέρασμα

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

# 8 Τεστ του Fermat

## 8.1 Εκφώνηση

Ικανοποιούν το τεστ ου Fermat οι αριθμοί

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

## 8.2 Λύση

**Βιβλιοθήκες**

- gmpy2

Εκτέλεση αρχείου:

```
python fermat.py
```

Η υλοποίηση ακολουθεί τον αλγόριθμο που δίνεται από το βιβλίο [1] σελίδα 122, προσθέτοντας τον έλεγχο για το εάν ο αριθμός είναι άρτιος δεν μπορεί ναι είναι και πρώτος.

### 8.2.1 Συμπέρασμα

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

# 9 Trial Division

## 9.1 Εκφώνηση

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

## 9.2 Λύση

### 9.2.1 Λεπτομέρειες για την υλοποίηση

Εκτέλεση αρχείου:

```
python trial_division.py
```

Ακολουθήθηκε πιστά ο αλγόριθμος δόθηκε στο βιβλίο [1] στην σελιδα 134.

### 9.2.2 Συμπέρασμα

Οι αριθμοί παραγοντοποιούνται ως εξης

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

και για

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

# 10 Αλγόριθμος του Lehman

## 10.1 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;

## 10.2 Λύση

**Βιβλιοθήκες**

- gmpy2
- sympy
- multiprocessing

Εκτέλεση αρχείου:

```
python lehman.py
```

Υλοποιήθηκε ο αλγόριθμος όπως παρουσιάζεται στο [1] στην σελίδα 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.

### 10.2.1 Συμπέρασμα

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

# 11 Αλγόριθμος του Pollard

## 11.1 Εκφώνηση

Σε αυτή την άσκησή θα δούμε πως μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο του Pollard για παραγοντοποιήση. Θεωρούμε ένα φυσικό $N$ και ζητάμε ένα διαιρέτη του. Έστω $f(x) = (x^2 + 1) \mod N$ και ας είναι το $x_0$ τυχαία τιμή από το $\{2, 3, ..., N - 1\}$. 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 $N = 2^{257} - 1$.

## 11.2 Λύση

### 11.2.1 Λεπτομέρειες για την υλοποίηση

Εκτέλεση αρχείου:

```
python pollard.py
```

Ακολουθήθηκε πιστά ο αλγόριθμος δόθηκε στο βιβλίο [1] στην σελίδα 150, προσθέτοντας την συνθήκη της εκφώνησης.

### 11.2.2 Συμπέρασμα

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

# 12 RSA

## 12.1 Εκφώνηση

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

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

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 $\phi(N)$.

## 12.2 Λύση

### 12.2.1 Λεπτομέρειες για την υλοποίηση

**Βιβλιοθήκες**

- gmpy2
- numpy
- fpow

Εκτέλεση αρχείου:

```
python rsa.py
```

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

$$m_i = c_i^d \mod N, \forall c_i \in C \tag{9}$$

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 [2] 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

$$e \cdot d \equiv 1 \mod \phi(N) \Rightarrow d = e^{-1} \mod \phi(N) \tag{10}$$

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

### 12.2.2 Συμπέρασμα

Εκτελώντας την σχέση (9) για κάθε block, επιστρέφεται το μήνυμα

```
welcowe to the real world
```

# 13 Wiener Attack

## 13.1 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 $m$.

## 13.2 Λύση

### 13.2.1 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 $\phi_i, p, q$ 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.

### 13.2.2 Συμπέρασμα

Εκτελώντας λοιπόν την επίθεσή του Wiener, το αρχικό μήνυμα είναι

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

# 14 Naive RSA

## 14.1 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;

## 14.2 Λύση

### 14.2.1 Λεπτομέρειες για την υλοποίηση

**Βιβλιοθήκες**

- fpow

Εκτέλεση αρχείου:

```
python signature.py
```

Για να γίνει επαλήθευση της υπογραφής θα πρέπει να ισχύει

$$a = m, \text{ όπου } a = s^e \mod N. \tag{11}$$

### 14.2.2 Συμπέρασμα

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

# 15 gpg

## 15.1 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

## 15.2 Λύση

### 15.2.1 Συμπέρασμα

d41d8cd98f00b204e9800998ecf8427e

# Αναφορές

[1] Κ. Δραζιώτης, *Εισαγωγή στην Κρυπτογραφία*. Κάλλιπος, Ανοικτές Ακαδημαϊκές Εκδόσεις, 2022. https://dx.doi.org/10.57713/kallipos-17.

[2] H. Böck, “Fermat factorization in the wild.” Cryptology ePrint Archive, Paper 2023/026, 2023. https://eprint.iacr.org/2023/026.