

# ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ ΣΤΗ ΚΡΥΠΤΟΓΡΑΦΙΑ

## ΠΑΡΑΤΗΡΗΣΕΙΣ

- Την εργασία την υλοποίησα στο περιβάλλον **NetBeans** σε λειτουργικό σύστημα **Ubuntu Linux** (τρέχει σε VM με Windows host) ώστε να γίνει ευκολότερα η εγκατάσταση της GNUPG βιβλιοθήκης (για αυτό ενώ η πρώτη εργασία με το CRT έγινε σε Windows και εδώ χρησιμοποιώ άλλο λειτουργικό).
- Κάποιοι παράμετροι είναι σταθεροί. Για παράδειγμα το **e** στον RSA το έθεσα με **65537** καθώς και το **q** στον ElGamal ίσο με  $(p-1)/2$ . Αυτές δεν είναι απαραίτητο να είναι τυχαίες, αρκεί να είναι ασφαλείς και να ικανοποιούν κάποιες συνθήκες πράγμα που συμβαίνει εδώ.
- Η μετατροπή των string αγγλικού αλφαβήτου σε ακέραιο έγινε με χρήση του πίνακα ASCII: Κάθε γράμμα αντιστοιχίζεται στην αντίστοιχη τιμή του στον πίνακα ASCII ως τριψήφιος αριθμός ώστε να χωρέσουν όλα τα γράμματα. Αυτή η μετατροπή δεν είναι πολύ καλή διότι μειώνεται ο αριθμός των πιθανών μηνυμάτων για κάποιο συγκεκριμένο μέγεθος  $N$  (RSA, Rabin) και  $p$  (ElGamal) αφού χρειαζόμαστε τρία ψηφία για ένα γράμμα. Μια πιο απλή μέθοδος και πιο αποτελεσματική (δηλαδή για ίδιο μέγεθος  $N$  ή  $p$  θα μπορούμε να κρυπτογραφήσουμε παραπάνω δεδομένα) είναι η αντιστοίχιση των a-z στο 00,01,...25 και A-Z στο 26,27,...52 (δύο αριθμοί για κάθε γράμμα).

**Ζήτημα 1 (3 μονάδες). Υλοποιήστε τον αλγόριθμο κρυπτογράφησης RSA.**

**1. Υλοποιήστε έναν αλγόριθμο που θα μετατρέπει ένα μήνυμα σε έναν ακέραιο και το αντίστροφο. Υποθέστε ότι τα μηνύματα αποτελούνται μόνο από χαρακτήρες του αγγλικού αλφαβήτου.**

**2. Δημιουργήστε δύο πρώτους αριθμούς  $p$  και  $q$  μεγέθους 512 bits. Υπολογίστε στη συνέχεια το  $N$ , ένα δημόσιο κλειδί  $e$  και το αντίστοιχό του ιδιωτικό  $d$ . Με τις παραμέτρους που έχετε δημιουργήσει, κρυπτογραφήστε και αποκρυπτογραφήστε ένα μήνυμα της επιλογής σας για να ελέγξετε την ορθότητα της υλοποίησής σας.**

1) Ο αλγόριθμος που έκανα είναι πολύ απλός, απλώς παίρνει την **ASCII** τιμή του αριθμού ως τριψήφιο αριθμό. (δηλαδή το 'α' στο ASCII είναι το 97 που είναι διψήφιο, αυτό θα γίνει 097 κτλ). Έτσι οποιοδήποτε μήνυμα γραμμένο στα αγγλικά μεταφράζεται σε αριθμό, για παράδειγμα το

**"RSA Encryption" → 082 083 065 032 069 110 099 114 121 112 116 105 111 110.**

2) Η διαδικασία αυτή είναι η εφαρμογή του RSA αλγορίθμου. Χάρis στη βιβλιοθήκη GMP μπορούμε και φτιάχνουμε δύο ψευδοτυχαίους αριθμούς μεγέθους 512 bits. Πρέπει όμως να είναι και prime, οπότε κάθε φορά ελέγχουμε αν είναι prime οι αριθμοί (πρέπει και οι δύο να είναι) χάρη στη μέθοδο **Miller-Rabin** που παρέχει η GMP. Αν δεν είναι prime τότε ξαναδημιουργούμε τυχαίους μέχρι το τεστ Miller-Rabin να πετύχει. Η μέθοδος αυτή έχει κάποιο ποσοστό αποτυχίας αλλά είναι γενικά πολύ μικρό (δηλαδή μπορεί να αναγνωρίσει έναν composite αριθμό ως prime). Στη συνέχεια υπολογίζουμε το  $N$  ως

$N = pq$  ( $p, q$  οι δύο prime). Το δημόσιο κλειδί είναι το ζευγάρι  $(e, N)$  όπου  $e$  είναι ένας αριθμός στο σύνολο  $[3, \phi(n)]$  αλλά συνήθως είναι το 65537. Το ιδιωτικό κλειδί είναι το  $(d, N)$  όπου  $d$  βρίσκεται (με τον επεκταμένο αλγόριθμο του Ευκλείδη) από τη σχέση  $ed = 1 \pmod{\phi(n)}$ . Το σημαντικό εδώ είναι πως το  $\phi(n)$  είναι κρυφό (μόνο αν κάποιος βρει τους factors  $p, q$  του  $N$  μπορεί να το υπολογίσει ως  $\phi(n) = (p-1)(q-1)$ , αφού  $p, q$  είναι πρώτοι και για δύο πρώτους ισχύει το  $\phi(pq) = \phi(p)\phi(q)$ ). Η κρυπτογράφηση γίνεται ως:

$c = m^e \pmod N$ ,  $c$ =το ciphertext,  $m$ =plaintext,  $e$ =το 65537 συνήθως και  $N=pq$ .

Η αποκρυπτογράφηση γίνεται ως:

$m = c^d \pmod N$ .

Βέβαια το πρόγραμμα δεν είναι ασφαλής αφού η απλή μετατροπή  $ASCII \leftrightarrow decimal$  δεν έχει τυχαιότητα. Αυτό που έκανα εγώ είναι το "textbook RSA" ενώ υπάρχουν τρόποι για να γίνει ασφαλές όπως η χρησιμοποίηση padding (OAEP).

Παρακάτω ακολουθεί εικόνα, η οποία είναι screenshot μιας εκτέλεσης του προγράμματος (θέλει zoom για να φανεί καλά).

```
p = 9817829400854087048144405744098025820664137490507800482388375570814506420930755518750054310308718750320883069229925367441583806543519612960257651768417369
q = 2093532303813002968577737307984491583142415081149051370388865406281658004957466421971914127583359287384710426324014458231473293359224467453325165719109067
n = 2055394300401309147274839046738016357764391722774020070213421406826893662551536360837978810949649843890160167831991923261598031979002754357646364123293205717318584
5227495399629233447182309756390373473032493111630432942078607139885981015187883083803635104778478597810517307291771276015049656633588399288184723
phi = 20553943004013091472748390467380163577643917227740200702134214068268936625515363608379788109496498438901601678319919232615980319790027543576463641232932045261824
140560405382907090395099792352583820901375641258853191964982442713997759074465914645911557067072885102256577481618714176112305576220005581800658288
Public key is n and e = 65537
private key = 820878136300164267829953113650622398769018858841375121195143077990861875331857007622766641710730351007538523777935465453851517295854526337138348330913473
8432991822253506423733313761712345164357979893382456235549742076864221059800987930711977202833347493378788999595093748628839050392350674458434565177692449
Plaintext message = rsa
Encoded characters: 114115097
Encrypted Ciphertext = 120289153166273708615889197069851913831260918213484788549772499247209604511607852849633203345522711804974161051882088782439608216891735545216087
00116108433930310964288894389480811206823794754912115907822899015535641132734702603859690501780750314908153861525256330178116077321030845947638421903192130298214646
Decrypted (and encoded) Plaintext = 114115097
Decoded plaintext = rsa
RUN FINISHED; exit value 0; real time: 3s; user: 0ms; system: 3s
```

**Ζήτημα 2 (3 μονάδες). Υλοποιήστε τον αλγόριθμο κρυπτογράφησης ElGamal.**

**1. Δημιουργήστε έναν πρώτο αριθμό  $p$  μεγέθους 200 bits και υπολογίστε έναν γεννήτορα του σώματος  $Z_p^*$  (εδώ δεν είναι απαραίτητο να υλοποιήσετε έναν αλγόριθμο παραγοντοποίησης).**

**2. Δημιουργήστε ένα δημόσιο κλειδί και το αντίστοιχό του ιδιωτικό κλειδί. Με τις παραμέτρους που έχετε δημιουργήσει, κρυπτογραφήστε και αποκρυπτογραφήστε ένα μήνυμα της επιλογής σας για να ελέγξετε την ορθότητα της υλοποίησής σας.**

Αντίστοιχα, απλώς εφαρμόζουμε τον αλγόριθμο. Δημιουργούμε έναν τυχαίο  $p$  200 bits και στη συνέχεια θέτουμε τον γεννήτορα  $g = (p-1)/2$ . Εφόσον δε χρειάστηκε να υλοποιήσουμε κάποιον αλγόριθμο θέτουμε πάντα τον συγκεκριμένο γεννήτορα (η οποία είναι και μια συνήθως τιμή στον ElGamal αλγόριθμο).

Για το δημόσιο κλειδί δημιουργούμε ένα τυχαίο  $a$  στο διάστημα  $[0, p-2]$ . Το δημόσιο κλειδί είναι το  $g^a \bmod p$ , ενώ το ιδιωτικό το  $a$ .

- Για την κρυπτογράφηση δεχόμαστε το public key καθώς και τα  $p, g$ . Εφόσον δε γνωρίζουμε το  $a$  δε μπορούμε να βρούμε εύκολα το private key (πρόβλημα διακριτού λογαρίθμου). Μετατρέπουμε το μήνυμα σε ακέραιο και επιλέγουμε ένα τυχαίο  $k$  στο διάστημα  $[1, p-2]$ . Υπολογίζουμε τις δύο τιμές  $c1 = g^k \bmod p$  και  $c2 = \text{plaintext} * (\text{public\_key})^k \bmod p$  και τις στέλνουμε. Είναι σημαντικό να επιλέγουμε διαφορετικό  $k$  σε κάθε κρυπτογράφηση, οπότε στη συνάρτηση κρυπτογράφησης παράγεται ξανά μια τυχαία τιμή για το  $k$ .
- Για την αποκρυπτογράφηση λαμβάνουμε τα ciphertext  $c1$  και  $c2$ . Αρχικά υπολογίζουμε το  $\text{temp} = c1^{(p-1-\text{private\_key})} \bmod p$ . Στη συνέχεια το plaintext είναι το  $\text{plaintext} = \text{temp} * c2 \bmod p$ .

Οποιαδήποτε άλλη διευκρίνιση χρειάζεται την έχω γράψει και στα σχόλια στον κώδικα αναλυτικά για κάθε βήμα. Παρακάτω είναι μια εκτέλεση του αλγορίθμου για το plaintext "elgamal". Όπως και στον RSA έτσι και εδώ γίνεται η μετατροπή σε ακέραιο με τον ίδιο τρόπο (και αντίστροφα).

```
p = 279587757781705892984591167517576441332835823382783127099557
g = 139793878890852946492295583758788220666417911691391563549778
Public key = 254232183000806702648723016805028777969814846812341912008927
Private key = 5362405689416016916366662623941537417359342387705425925766
Plaintext message = elgamal
Encoded characters: 101108103097109097108
Encrypted Ciphertext c1 = 208910931268806330475947826797602886491867943544860524851100
Encrypted Ciphertext c2 = 259837068639704143830780638646493862466868967028445205190992
Decrypted (and encoded) plaintext = 101108103097109097108
Decoded plaintext = elgamal
RUN FINISHED; exit value 0; real time: 0ms; user: 0ms; system: 0ms
```

### Ζήτημα 3 (3 μονάδες). Υλοποιήστε τον αλγόριθμο κρυπτογράφησης Rabin.

Αναλυτικότερα:

1. Δημιουργήστε δύο πρώτους αριθμούς  $p$  και  $q$  μεγέθους 200 bits που να είναι ισότιμοι με  $3 \bmod 4$ .

2. Με τις παραμέτρους που έχετε δημιουργήσει, κρυπτογραφήστε και αποκρυπτογραφήστε ένα μήνυμα της επιλογής σας για να ελέγξετε την ορθότητα της υλοποίησής σας. Χρησιμοποιήστε την παρατήρηση 8.12 του βιβλίου (θα χρειαστεί να υλοποιήσετε και τον αλγόριθμο 2.107) καθώς και έναν τρόπο για να ξεχωρίζετε το σωστό από τα τέσσερα μηνύματα που προκύπτουν από την αποκρυπτογράφηση.

Αντίστοιχα με τον RSA, η δημιουργία των παραμέτρων (δηλαδή  $n, p, q$ ) είναι η ίδια, η μόνη διαφορά είναι πως κάθε φορά που το τεστ Miller-Rabin επιτύχει (δηλαδή  $p, q$  είναι με μεγάλη πιθανότητα prime) ελέγχουμε αν είναι ισότιμοι με  $3 \bmod 4$  (εύκολα γίνεται αυτό, απλώς ελέγχουμε αν  $p \bmod 4 = q \bmod 4 = 3 \bmod 4 \rightarrow p \bmod 4 = q \bmod 4 = 3$ ).

Η κρυπτογράφηση στον αλγόριθμο Rabin είναι η:  $c = m^2 \bmod n$ ,

$m = \text{plaintext}$ ,  $c = \text{ciphertext}$ ,  $n = pq$ .

Η αποκρυπτογράφηση τώρα θέλει πιο πολύ δουλειά. Επειδή έχουμε 4 πιθανά plaintext από κάποιο ciphertext που έχει κρυπτογραφηθεί από τον αλγόριθμο αυτό, βάζουμε κάποιο redundancy. Δηλαδή (εγώ συγκεκριμένα) πρόσθεσα 12 άσους στο τέλος. Είναι σχεδόν αδύνατο να έχουν 12 άσους στο τέλος (ως πιο ασήμαντα ψηφία δηλαδή) κάποια από τα υπόλοιπα τρία πιθανά plaintext, οπότε με μεγάλη πιθανότητα επιλέγουμε αυτό.

Αρχικά όμως πρέπει να βρούμε πως υπολογίζονται αυτά τα plaintexts, δηλαδή να δούμε πως γίνεται η αποκρυπτογράφηση.

Αρχικά χρησιμοποιούμε τον Επεκταμένο Αλγόριθμο του Ευκλείδη για να βρούμε τους  $a, b$  οι οποίοι ικανοποιούν τη σχέση:  $ap + bq = 1$ .

Εφόσον έχουμε τα  $a, b$  υπολογίζουμε τα  $r, s, x$  και  $y$  ως

- $r = c^{((p+1)/4)} \bmod p$ .
- $s = c^{((q+1)/4)} \bmod q$ .
- $x = (aps + bqr) \bmod n$ .
- $y = (aps - bqr) \bmod n$ .

Τα τέσσερα πιθανά plaintext είναι τα  $x, y, -x \bmod n, -y \bmod n$ .

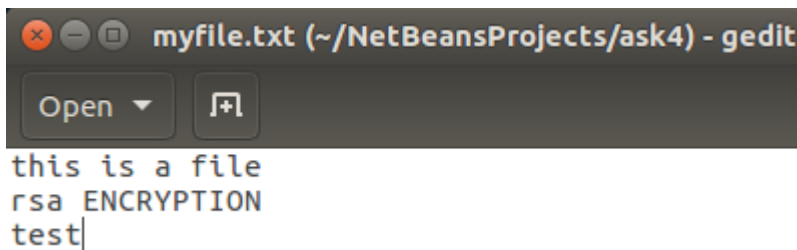
Άρα σε κάθε ένα από αυτά ελέγχουμε το redundancy και αν βρούμε πως κάποιο έχει το redundancy τότε απλώς το αφαιρούμε. Το αποτέλεσμα είναι το plaintext. Ακολουθεί εικόνα εκτέλεσης (zoom για να φανεί καλύτερα). Μόνο το  $-y \bmod n$  έχει το redundancy εδώ.

```
p = 1239410297438791359459214786726203837821070201671541287203999
q = 93224397133785725268941919325390519148983254232973361432467
n = 115543277780137374443039273959862308428624717990931158028443629204094461601607566954175590279812891469214385523890835533
Plaintext message = rabin
Encoded characters (plus redundancy): 114097098105110111111111111
Encrypted Ciphertext = 13018147796007121307518802099101879433432320987654321
a = -21742001556318901971095184746858470206475572572496590024960
b = 289058030347571319813512448836689173054089534121197753344923
d (MUST BE 1) = 1
1) x = 91778742901385173413687674504952597811828529632323674541760419037811357715789201997888649005296317757404178377492037412
2) y = 115543277780137374443039273959862308428624717990931158028443629204094461601607566954175590279698794371109275412779724422
3) -x MOD n = 23764534878752201029351599454909710616796188358607483486683210166283103885818364956286941274516573711810207146398798121
3) -y MOD n = 11409709810511011111111111111
Decrypted and decoded (no redundancy) plaintext: rabin
RUN FINISHED; exit value 0; real time: 450ms; user: 0ms; system: 450ms
```

**Ζήτημα 4 (1 μονάδα).** Υλοποιήστε μια συνάρτηση που να κρυπτογραφεί ένα αρχείο κειμένου. Μπορείτε να χρησιμοποιήσετε όποιον αλγόριθμο επιθυμείτε από τα ζητήματα 1, 2 και 3.

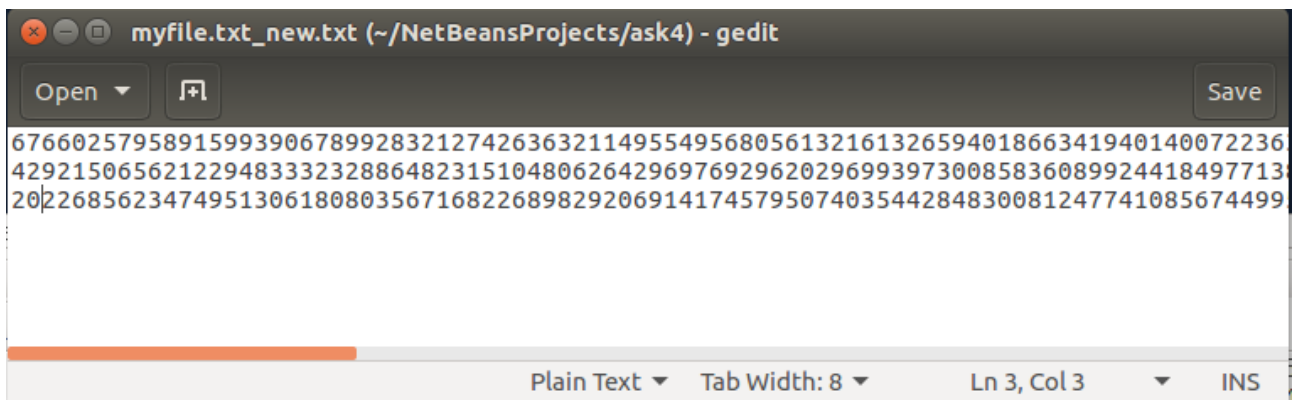
Χρησιμοποίησα τον RSA. Η συνάρτηση υλοποιείται εύκολα, απλώς ανοίγουμε το αρχείο και παράγουμε ένα νέο αρχείο με τα κρυπτογραφημένα δεδομένα, διαβάζοντας γραμμή-γραμμή και κρυπτογραφώντας κάθε γραμμή. Θα μπορούσαμε επίσης να διαβάσουμε όλο το αρχείο και να κρυπτογραφήσουμε όλο το κείμενο αλλά γενικά οι public key αλγόριθμοι κρυπτογραφίας έχουν μικρό όριο για το μήκος ενός μηνύματος (στον RSA πρέπει να είναι μικρότερο από το  $N$  αφού οι πράξεις κρυπτογράφησης/αποκρυπτογράφησης δουλεύουν στο  $\text{mod } N$ ).

Ένα παράδειγμα αρχικού κειμένου και κρυπτογραφημένου φαίνεται παρακάτω.



```
myfile.txt (~/.NetBeansProjects/ask4) - gedit
Open [Add]
this is a file
rsa ENCRYPTION
test|
```

Αρχικό αρχείο



```
myfile.txt_new.txt (~/.NetBeansProjects/ask4) - gedit
Open [Add] Save
676602579589159939067899283212742636321149554956805613216132659401866341940140072236
429215065621229483332328864823151048062642969769296202969939730085836089924418497713
202268562347495130618080356716822689829206914174579507403544284830081247741085674499
Plain Text Tab Width: 8 Ln 3, Col 3 INS
```

Το νέο κρυπτογραφημένο αρχείο. Έχει 3 γραμμές, κάθε γραμμή έχει κρυπτογραφημένα το κωδικοποιημένα σε ASCII γράμματα της κάθε γραμμής.