Κρυπτογραφία

Γέροντας Αλέξανδρος – 321/2015029

2η Άσκηση

Σημείωση: Όλα τα ζητήματα υλοποιήθηκαν σε λειτουργικό Windows 10 με το gnump library εγκαταστημένο από το Msys2. Στα linux δεν έγιναν αρκετές δοκιμές αλλά ο κώδικας τρέχει κανονικά.

Ζήτημα 1 – Αλγόριθμος κρυπτογράφησης RSA

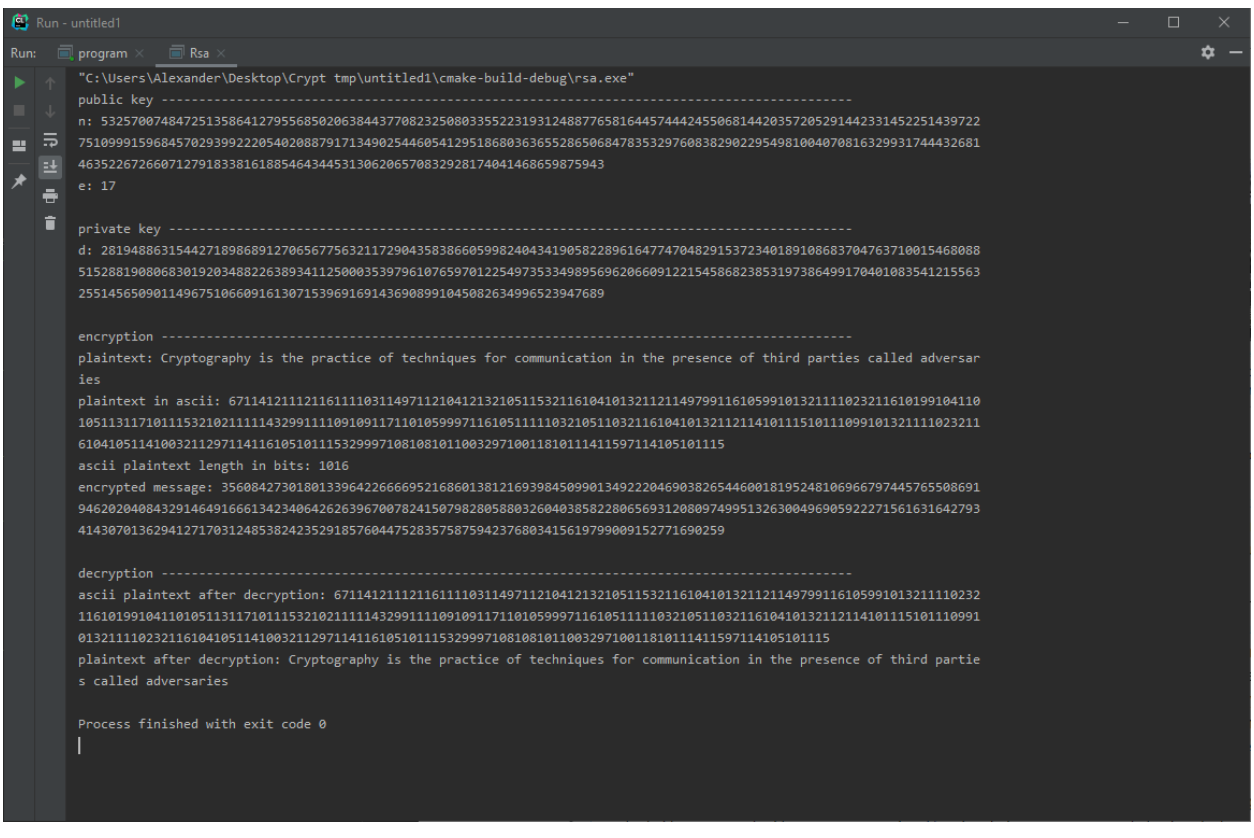
Αρχικά δημιουργούμε μία συνάρτηση (convert_plaintext_to_ascii) η οποία μετατρέπει το αρχικό μήνυμα σε έναν ακέραιο. Συγκεκριμένα η συνάρτηση παίρνει σαν όρισμα το plaintext και έναν ακέραιο mpz με όνομα ascii_plaintext. Για κάθε χαρακτήρα του plaintext, μετατρέπουμε τον χαρακτήρα σε ascii με την συνάρτηση sprintf και τον αποθηκεύουμε σε έναν προσωρινό δείκτη τύπου char. Αφού μετατρέψουμε όλους τους χαρακτήρες σε ascii, μετατρέπουμε τον δείκτη σε ακέραιο mpz και τον αποθηκεύουμε στην μεταβλητή ascii_plaintext.

Στην συνέχεια δημιουργούμε μια συνάρτηση η οποία δημιουργεί δημόσια και ιδιωτικά κλειδιά. Πρώτα δημιουργεί έναν τυχαίο αριθμό 512 bit (Τα bit δίνονται σαν όρισμα στην συνάρτηση) και με βάση τους επόμενους πρώτους από αυτόν δημιουργούμε τους ακέραιους p,q. Στη συνέχεια υπολογίζουμε το n και το ϕ, και βρίσκουμε έναν ακέραιο e ο οποίος να αντιστρέφεται στο Zϕ – γίνεται επανάληψη μέχρι να βρεθεί και υπολογίζεται και ο αντίστροφος d.

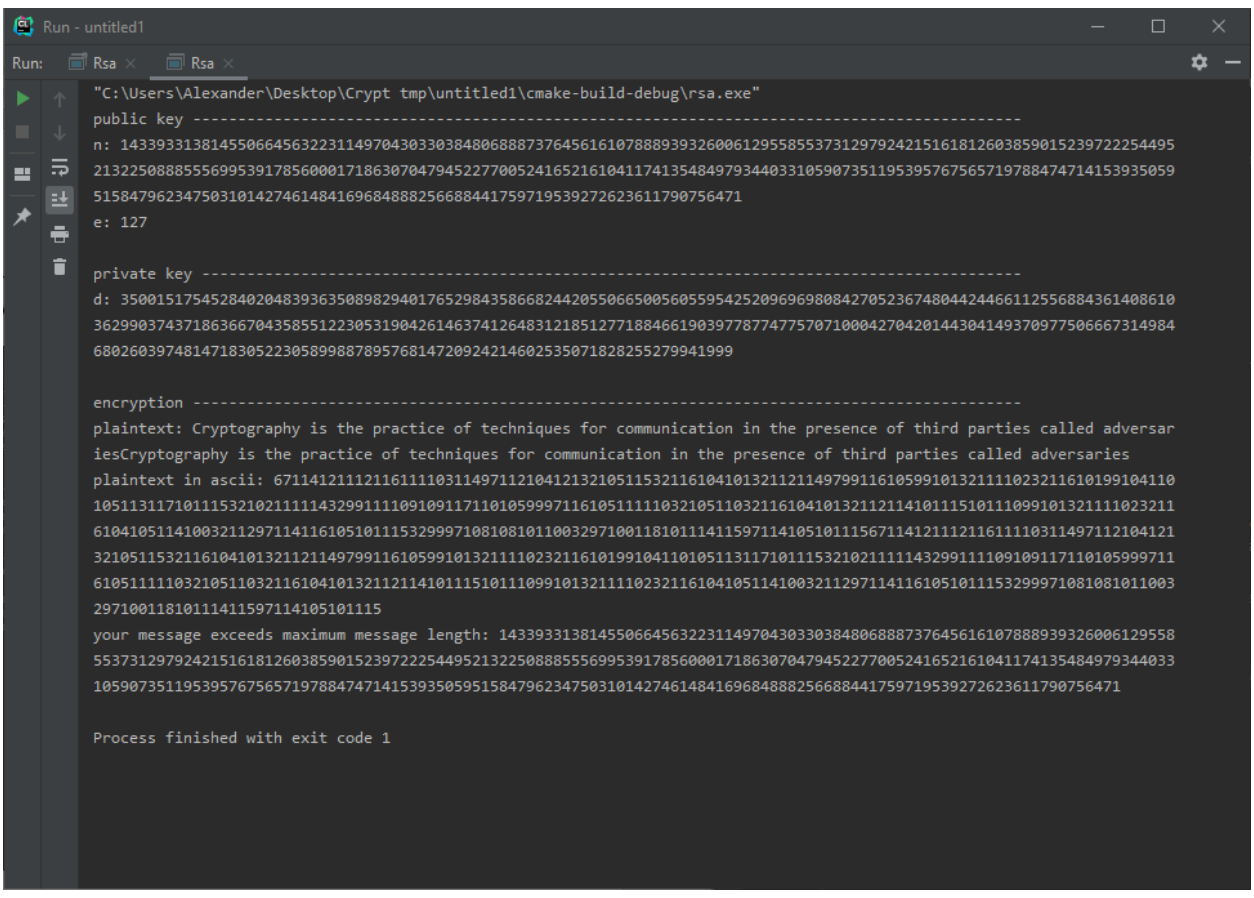
Για την κρυπτογράφηση του μηνύματος μετατρέπουμε το μήνυμα σε ακολουθία ascii (m) με την χρήση της συνάρτησης convert_plaintext_to_ascii. Ο υπολογισμός του ciphertext c γίνεται με τον τύπο c = m^e mod n.

Στην αποκρυπτογράφηση υπολογίζουμε το ascii_plaintext με τον τύπο ascii_plaintext = c^d mod n. Μετατρέπουμε το ascii_plaintext πίσω στο αρχικό μήνυμα με την συνάρτηση convert_ascii_to_plaintext η οποία ακολουθεί την αντίστροφη διαδικασία της convert_plaintext_to_ascii.

Κατά την εκτέλεση το πρόγραμμα τυπώνει τα ιδιωτικά και δημόσια κλειδιά το αρχικό κείμενο το αρχικό κείμενο σε ascii, το κρυπτοκείμενο και το αρχικό κείμενο μετά την αποκρυπτογράφηση.



Εάν δοθεί κείμενο του οποίου η ascii μορφή να ξεπερνάει το διάστημα 0, n-1 το πρόγραμμα τυπώνει μήνυμα αποτυχίας και τερματίζει.



Σημείωση: Οι συναρτήσεις convert_plaintext_to_ascii και convert_ascii_to_plaintext χρησιμοποιούνται σε όλα τα παρακάτω ζητήματα.

Ζήτημα 2 – Αλγόριθμος κρυπτογράφησης El Gamal

Με την συνάρτηση gen_keys δημιουργούμε το δημόσιο κλειδί p,a,a^d και το ιδιωτικό d. Συγκεκριμένα βρίσκουμε τον επόμενο πρώτο ενός τυχαίου αριθμού για να υπολογίσουμε το p. Καθώς το p είναι πρώτος η συνάρτηση ϕ(p) μας δίνει πάντα p-1. Για τον υπολογισμό του γεννήτορα του p κάνουμε επανάληψη με μία while μέχρι να βρεθεί ένας ακέραιος a τέτοιος ώστε a^ϕ(p) ≡ 1 mod p. Στη συνέχεια δημιουργούμε έναν ακέραιο d 6 bit και υπολογίζουμε το a^d.

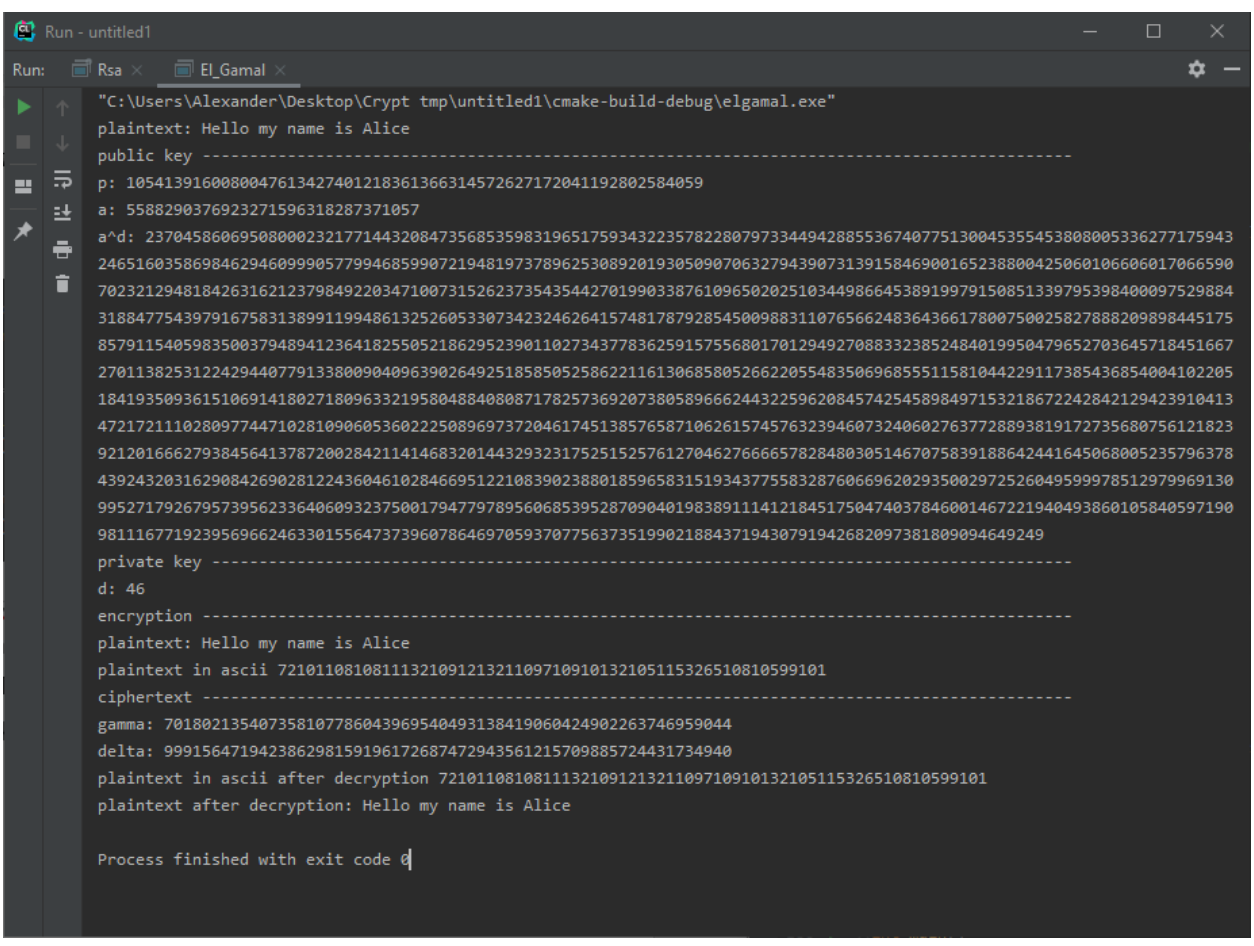
Για την κρυπτογράφηση του μηνύματος με την συνάρτηση encryptMsg μετατρέπουμε πρώτα το αρχικό μήνυμα σε ascii με την συνάρτηση convert_plaintext_to_ascii. Εάν το μήνυμα ascii ξεπερνάει το p στο δεκαδικό το πρόγραμμα τυπώνει μήνυμα αποτυχίας και τερματίζει. Δημιουργούμε έναν τυχαίο k 7 bit και υπολογίζουμε το κρυπτοκείμενο γ,δ ως εξής:

```
// γ = α^k mod p.
mpz_powm(gamma, a, k, p);

// tmp = α^dk
mpz_pow_ui(tmp, a_d, mpz_get_ui(k));
mpz_mul(tmp, tmp, ascii_plaintext);

// δ = m*a^dk mod p
mpz_mod(delta, tmp, p);
```

Στην συνάρτηση decryptMsg αποκρυπτογραφούμε το μήνυμα. Υπολογίζουμε το c = $γ^{p-1-d}$ mod p και το m = c*δ mod p για να πάρουμε το αρχικό κείμενο σε ascii. Τέλος με την συνάρτηση convert_ascii_to_plaintext μετατρέπουμε το αρχικό μήνυμα σε ascii στο αρχικό μήνυμα.



Ζήτημα 3 – Αλγόριθμος Κρυπτογράφησης Rabin

Για την δημιουργία των κλειδιών φτιάχνουμε μία random μεταβλητή 200 bit και παίρνουμε τους επόμενους πρώτους (p, q) μετά από αυτήν. Καθώς για τους p και q πρέπει να ισχύει ότι $p \equiv 3 \bmod 4$ και $q \equiv 3 \bmod 4$ αυτή η διαδικασία γίνεται σε επανάληψη μέχρι να τηρηθεί ο περιορισμός. Στη συνέχεια υπολογίζεται το n -> n = p*q. (συνάρτηση gen_keys)

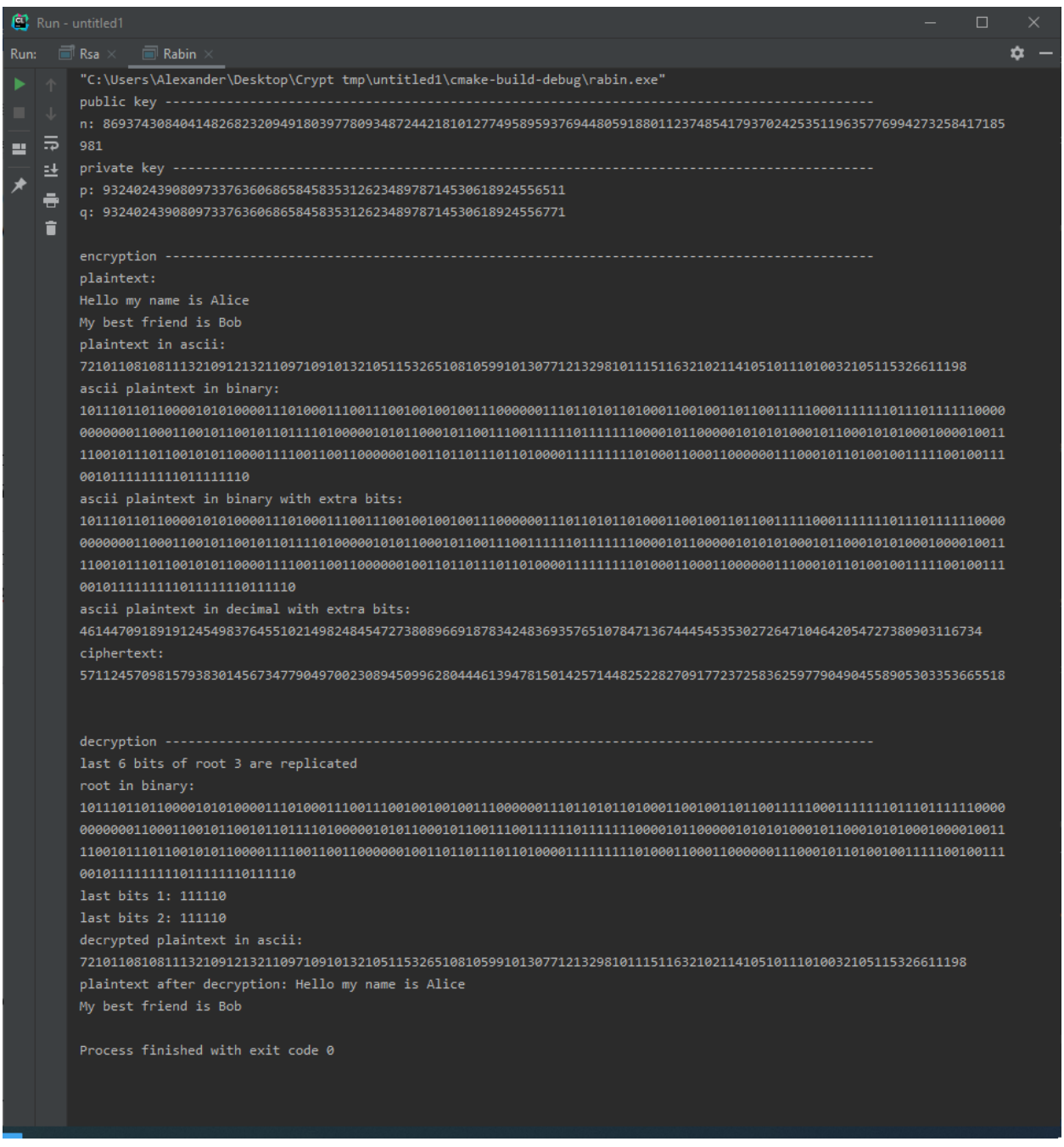
Για την κρυπτογράφηση του κειμένου με την συνάρτηση πρώτα μετατρέπουμε το αρχικό μήνυμα σε ascii με την συνάρτηση convert_plaintext_to_ascii. Μετατρέπουμε το ascii μήνυμα στο δυαδικό, και το τροποποιούμε έτσι ώστε να επαναλαμβάνονται τα τελευταία 6 bit του. Στην συνέχεια το μετατρέπουμε ξανά σε δεκαδικό, ελέγχουμε αν ξεπερνάει το n (αν ναι το πρόγραμμα τερματίζει) και υπολογίζουμε την τιμή $c = m^2 \bmod n$, όπου m το τροποποιημένο ascii plaintext. (συνάρτηση encryptMsg)

Στον υπολογισμό των ριζών βρίσκουμε αρχικά δύο αριθμούς a,b έτσι ώστε ap + bq =1 με την χρήση της έτοιμης συνάρτησης mpz_gcdext και υπολογίζουμε τις τέσσερις ρίζες με την χρήση των τύπων:

1. Βρίσκει ακεραίους a και b τέτοιους ώστε ap + bq = 1.
2. Υπολογίζει τις τιμές $r = c^{(p+1)/4} \bmod p$ και $s = c^{(q+1)/4} \bmod q$.
3. Υπολογίζει $x = (aps + bqr) \bmod n$.
4. Υπολογίζει $y = (aps - bqr) \bmod n$.
5. Οι 4 ρίζες του c mod n είναι οι x, -x mod n, y και -y mod n.

Αφού τις υπολογίζουμε τις καταχωρούμε σε έναν πίνακα. (συνάρτηση calculate_roots).

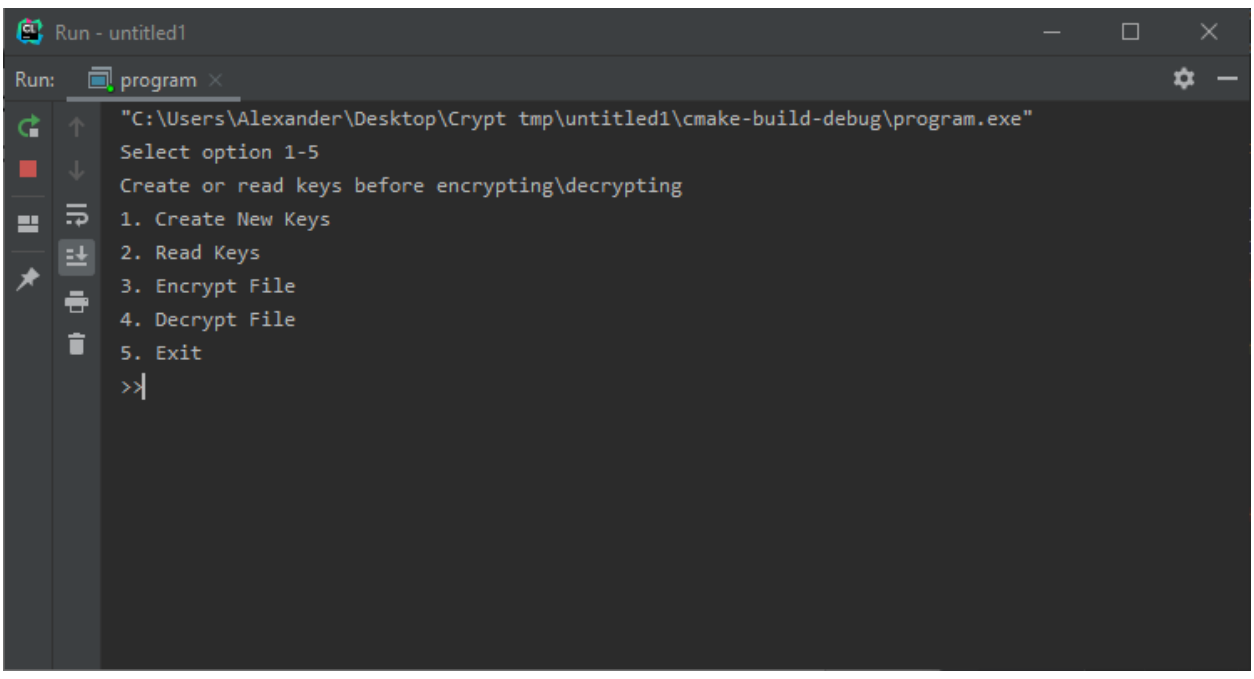
Τέλος για την αποκρυπτογράφηση του μηνύματος μετατρέπουμε τις ρίζες στο δυαδικό και ελέγχουμε τα τελευταία τους bit. Αυτό γίνεται σε επανάληψη μέχρι να βρεθεί μία ρίζα όπου τα τελευταία 6 bit της να επαναλαμβάνονται. Όταν βρεθεί αυτή η ρίζα αφαιρούμε τα bit και την μετατρέπουμε σε δεκαδικό. Μετατρέπουμε την ρίζα στο δεκαδικό και με την συνάρτηση convert_ascii_to_plaintext παίρνουμε το αρχικό μήνυμα.(συνάρτηση decryptMsg)



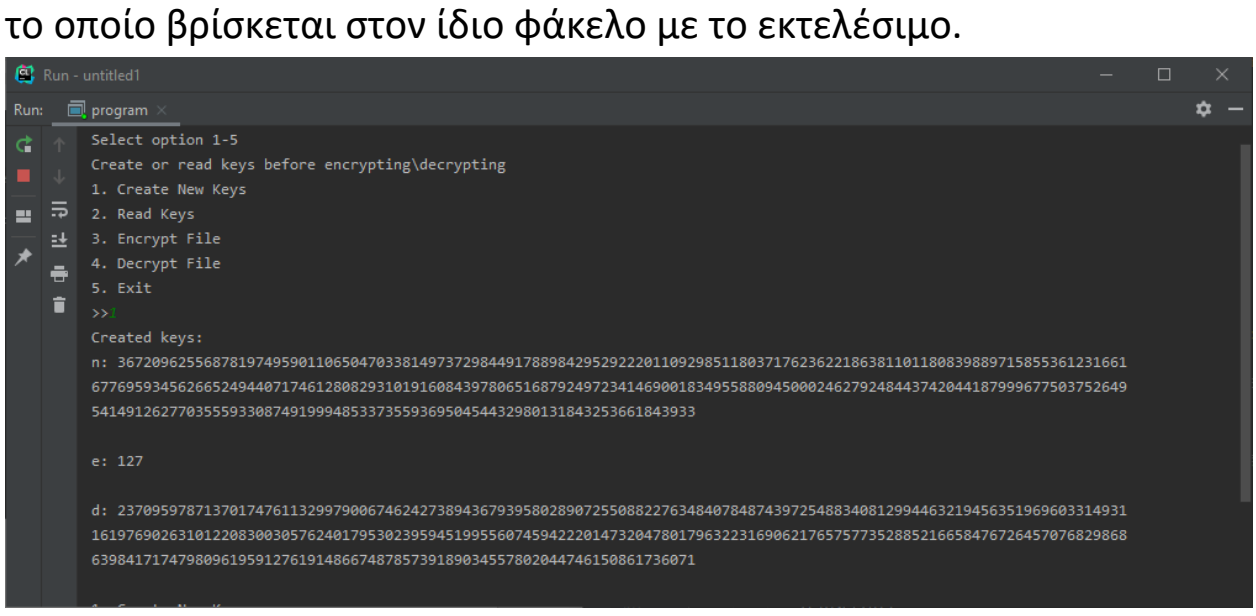
Ζήτημα 4 – Κρυπτογράφηση αρχείου κειμένου

Στο 4ο Ζήτημα για το πρόγραμμα που κρυπτογραφεί ένα αρχείο κειμένου επέλεξα τον αλγόριθμο RSA. Επίσης χρειάστηκε να υλοποιήσω ακόμα μία συνάρτηση. Η getLine δέχεται σαν όρισμα έναν δείκτη σε αρχείο και επιστρέφει μία γραμμή κειμένου. Διαβάζει από το αρχείο έναν-έναν τους χαρακτήρες και τους καταχωρεί στον δείκτη line. Με την χρήση της realloc προσαρμόζει το μέγεθος του line δίνοντας του περισσότερη μνήμη όποτε χρειάζεται. Η συνάρτηση getLine καλείται επαναληπτικά από το πρόγραμμα για να διαβάσει το plaintext ή το ciphertext και τα αρχείο με τα κλειδιά.

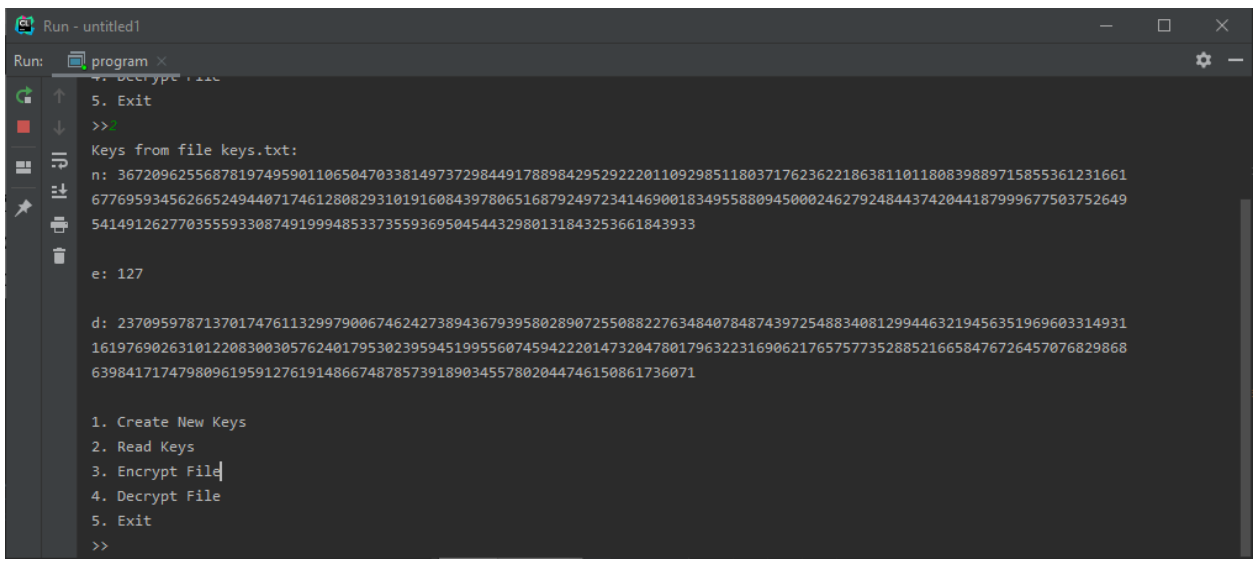
Το πρόγραμμα δίνει στον χρήστη τέσσερις επιλογές:



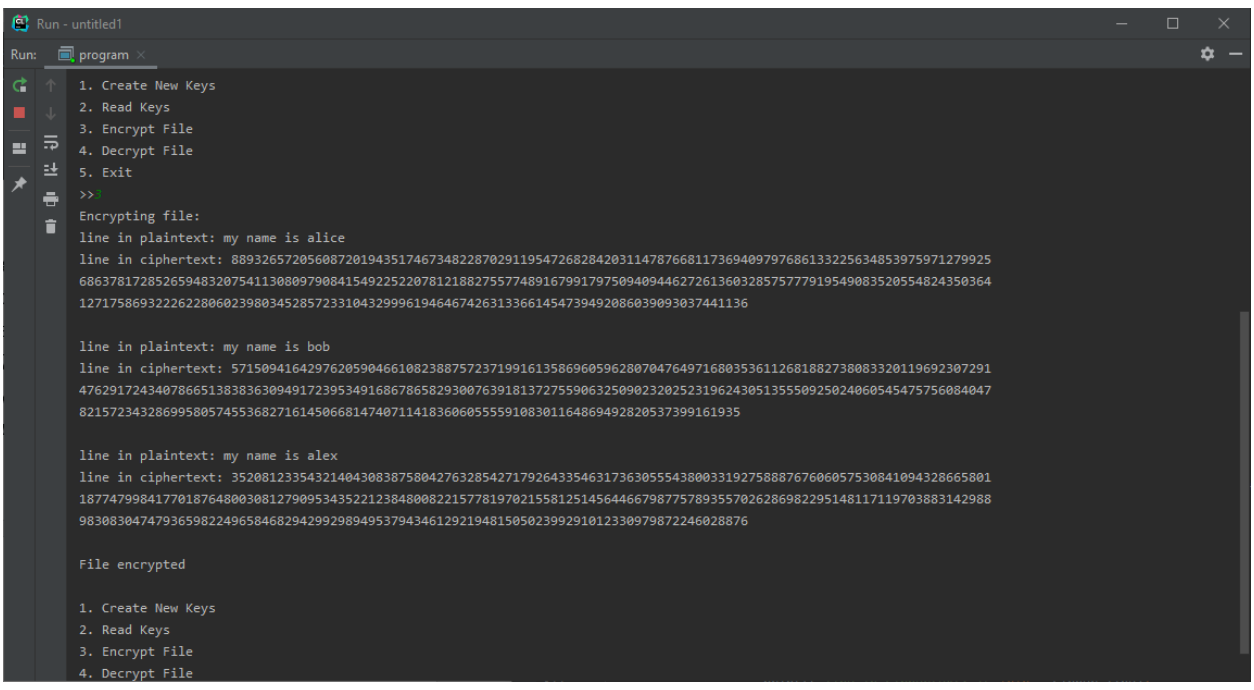
- Create New Keys
  Το πρόγραμμα δημιουργεί τα ιδιωτικά και δημόσια κλειδιά n, e, d. Μετά την δημιουργία τους καταχωρούνται στο αρχείο keys.txt το οποίο βρίσκεται στον ίδιο φάκελο με το εκτελέσιμο.



- Read Keys
  Το πρόγραμμα διαβάζει από το αρχείο keys.txt τα κλειδιά που έχουν δημιουργηθεί στην 1η επιλογή και τα καταχωρεί στις αντίστοιχες μεταβλητές n,e,d.

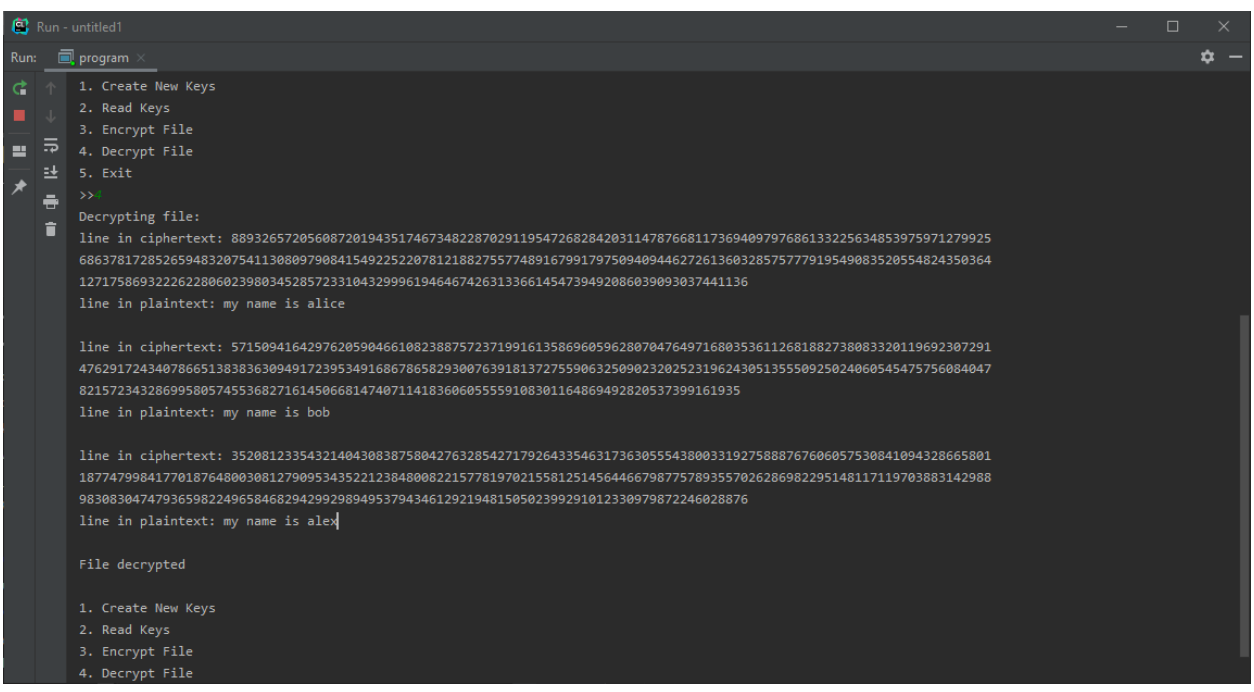


- Encrypt File
  Το πρόγραμμα διαβάζει το αρχείο plaintext.txt, κρυπτογραφεί την κάθε γραμμή του και την καταχωρεί στο αρχείο ciphertext.txt. Το αρχείο plaintext.txt πρέπει να δημιουργηθεί από τον χρήστη στον ίδιο φάκελο με το εκτελέσιμο (Στα linux πρέπει να ονομαστεί “.\plaintext.txt”). Επίσης ο χρήστης θα πρέπει πρώτα να έχει δημιουργήσει τα κλειδιά έχοντας επιλέξει μία από τις παραπάνω επιλογές.



- Decrypt File
  Το πρόγραμμα αποκρυπτογραφεί το αρχείο ciphertext.txt και καταχωρεί το αποκρυπτογραφημένο κείμενο στο αρχείο decrypted_plaintext.txt το οποίο επίσης βρίσκεται στον ίδιο φάκελο με το εκτελέσιμο. Για την αποκρυπτογράφηση απαιτούνται φυσικά τα ίδια κλειδιά που δημιουργήθηκαν με την 1η ή 2η επιλογή αλλιώς η αποκρυπτογράφηση δεν γίνεται σωστά.

Αποκρυπτογράφηση με σωστά κλειδιά



Αποκρυπτογράφηση με λάθος κλειδιά