Κρυπτογραφία

Προγραμματιστική Εργασία

Κοτζιάς Αριστείδης
icsd11186@icsd.aegean.gr

27 Απριλίου 2018

Περιεχόμενα

Αρχεία

rsa.c : Ζήτημα 1

elgamal.c: Ζήτημα 2

rabin.c: Ζήτημα 3

elgamal_file.c: Ζήτημα 4

Το αρχείο function.c περιέχει συναρτήσεις που χρησιμοποιούν οι υλοποιήσεις των αλγορίθμων. Το αρχείο function.h περιέχει δηλώσεις των συναρτήσεων αυτών.

Με το εκτελέσιμο αρχείο compile εκτελούνται τα απαραίτητα για να τρέξουν τα προγράμματα.

Χρειάζεται ο compiler gcc και λειτουργικό Linux.

Ζήτημα 1

Μετατροπή αριθμών σε χαρακτήρες

Για μετατροπή από string σε mpz και αντίστροφα υλοποίηθηκαν οι συναρτήσεις str2mpz(functions.c:45) και mpz2str(functions.c:62). Γίνεται μετατροπή βάσης σύμφωνα με το αλφάβητο το οποίο είναι δηλωμένο στο functions.h.

```
#define CHARSET "\n abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
#define BASE 54
```

Στο αλφάβητο περιλαμβάνεται το κενό και το newline.

rsa.c

- Δημιουργούμε 2 πρώτους μεγέθους 512 bits και τους συνενώνουμε έτσι ώστε να υπολογίσουμε εύκολα την τάξη του πρώτου αριθμού e που θα χρησιμοποιήσουμε για κρυπτογράφηση.
- Με την συνάρτηση set_prime(functions.c:26) υπολογίζουμε τον e ο οποίος είναι σχετικά πρώτος με το fi.

- Με την συνάρτηση mpz_invert υπολογίζουμε τον d οποίος είναι ο αντίστροφος του e.
- Με την συνάρτηση rsa_encrypt(functions.c:115) κρυπτογραφούμε.
- Με την συνάρτηση rsa_decrypt(functions.c:132) αποκρυπτογραφούμε.

Ζήτημα 2

elgamal.c

- Δημιουργούμε τα κλειδία με την συνάρτηση elgamal_genkeys(functions.c:494).
- Με την συνάρτηση elgamal_encrypt(functions.c:444) κρυπτογραφούμε.
- Με την συνάρτηση elgamal_decrypt(functions.c:476) αποκρυπτογραφούμε.

elgamal_genkeys()

- Δημιουργούμε safe_prime(functions.c:416) όπου $p = 2q + 1$ και q πρώτος για εύκολη παραγοντοποίηση της τάξης του p.
- Βρίσκουμε τον γεννήτορα a με την συνάρτηση get_generator(functions.c:376).
- Υπολογίζουμε το $a^d \mod p$.

Ζήτημα 3

rabin.c

- Δημιουργούμε τυχαίους πρώτους ισότιμους με 3 mod 4 με διπλό while.

```
int bits=200;
mpz_urandomb(p,stat,bits);
while (!mpz_probab_prime_p(p,10))
  while (!mpz_congruent_ui_p(p,3,4))
    mpz_nextprime(p,p);
```

Για ευκολία χρησιμοποιούμε προυπολογισμένους.

```
mpz_set_str(p,"6452532e37332ec325af793ea8125363cbf1dd7695f2c7fdcf",16);
mpz_set_str(q,"1bb2c0879d6cacd11c16f45d8b7eb42c7b5006c1485aa03b27",16);
```

- Υπολογίζουμε το n = pq
- Με την συνάρτηση rabin_encrypt(functions.c:154) κρυπτογραφούμε.
- Με την συνάρτηση rabin_decrypt(functions.c:245) αποκρυπτογραφούμε.

rabin_encrypt()

- Πριν κρυπτογραφήσουμε προσθέτουμε τα τελευταία N ψηφία του δυαδικού αριθμού
- Η σταθερά RABIN_RED(functions.h:5) καθορίζει πόσα bits θα χρησιμοποιείσουμε για redundancy (επιπλέον ψηφία)
- Βρίσκουμε τα τελευταία ψηφία με τη συνάρτηση lastNbits(functions.c:138) η οποία χρησιμοποιεί μάσκα με N άσσους και λογικό AND.
- Προσθέτουμε τα επιπλέον bits κάνοντας ολίσθηση αριστερα N bits και λογικό OR

rabin_decrypt()

- Υπολογίζουμε τα a,b με τη βοήθεια του επεκταμένου αλγόριθμου του Ευκλείδη(συνάρτηση extended_euclid(functions.c:180)) και στη συνέχεια τις 4 ρίζες
- Με τη συνάρτηση validate_rabin_root(functions.c:345) βρίσκουμε τη σωστή ρίζα, κάνοντας ολίσθηση δεξιά N bits και συγκρίνοντας τα τελευταία N bits πριν και μετά την ολίσθηση

Ζήτημα 4

Για επεξεργασία αρχείου επιλέχθηκε ο αλγόριθμος El Gamal. Χρειάζεται η τοποθεσία keys για load/save.

syntax

Κρυπτογράφηση: elgamal_file e file.txt

- Παράγει 2 αρχεία: file.txt.gamma file.txt.delta

Αποκρυπτογράφηση: elgamal_file d file.txt.gamma file.txt.delta

- Παράγει 1 αρχείο: decrypted.txt