Structure and Interpretation of Computer Programs
Second Edition
3η Εργασία

Κωδικοποίηση RSA

Κωδικοποίηση Δημόσιου-Κλειδιού (Public Key)
και
Ψηφιακές Υπογραφές (Digital Signatures)

Οι έννοιες του *δημόσιου κλειδιού κρυπτογράφησης* και των *ψηφιακών υπογραφών* εμφανίστηκαν το 1976 και διαδραματίζουν θεμελιώδη ρόλο στη προστασία των επικοινωνιών, σε έναν κόσμο που στηρίζεται άμεσα στην ψηφιακή πληροφορία. Υπάρχουν γρήγοροι αλγόριθμοι για ύψωση σε δύναμη και για έλεγχο αν ένας αριθμός είναι πρώτος, αλλά δεν υπάρχουν αλγόριθμοι που να παραγοντοποιούν ένα μεγάλο ακέραιο, σε αυτά βασίζεται η μέθοδος RSA (για την εφαρμογή κρυπτογράφησης δημόσιου κλειδιού). Σε αυτήν την εργασία θα υλοποιήσετε μια εκδοχή του συστήματος RSA. Με τον τρόπο αυτό, θα αποκτήσετε εμπειρία στις αναδρομικές συναρτήσεις, στη χρήση της δομής δεδομένων λίστας και αλγόριθμους που, αν και απλοί, έχουν τεράστια πρακτική σημασία. [1] Το πρώτο τμήμα της παρούσης εργασίας περιγράφει πώς λειτουργεί το σύστημα. Το δεύτερο κεφάλαιο περιέχει υπόβαθρο για την τελική εργασία, και το τμήμα 3 είναι η εργασία.

1. Το σύστημα RSA

Ο άνθρωπος χρησιμοποιεί μυστικούς κώδικες για χιλιάδες χρόνια. Το 1976, οι Whitfield Diffie και Martin Hellman από το Πανεπιστήμιο του Stanford επινόησαν μια καινούργια μέθοδο για κρυπτογράφηση και αποκρυπτογράφηση: *την κρυπτογράφηση δημόσιου κλειδιού*. [2]

Τα συστήματα κρυπτογράφησης συνήθως βασίζονται στην έννοια του *κλειδιού* για κρυπτογράφηση και αποκρυπτογράφηση. Ένα *κλειδί κρυπτογράφησης* ορίζει τη μέθοδο για την μετατροπή του αρχικού μηνύματος σε μία κωδικοποιημένη μορφή. Ένα αντίστοιχο *κλειδί αποκρυπτογράφησης* περιγράφει τον τρόπο αναίρεσης της κωδικοποίησης. Στα παραδοσιακά κρυπτογραφικά συστήματα, το κλειδί αποκρυπτογράφησης είναι πανομοιότυπο με το κλειδί κρυπτογράφησης, ή μπορεί να παραχθεί άμεσα από αυτό. Κατά συνέπεια, αν γνωρίζεις πώς να *κρυπτογραφήσεις* ένα μήνυμα με ένα συγκεκριμένο κλειδί, εύκολα μπορείς να *αποκρυπτογραφήσεις* μηνύματα που έχουν κρυπτογραφηθεί με το συγκεκριμένο κλειδί.

Η διορατικότητα των Diffie και Hellman ήταν στην παρατήρηση ότι μπορεί να οριστεί ένα κρυπτογραφικό σύστημα για το οποίο η γνώση του κλειδιού κρυπτογράφησης δεν προσφέρει καμία πληροφορία για την αποκρυπτογράφηση μηνυμάτων. Δηλαδή, δεν υπάρχει πρακτικός τρόπος εύρεσης του κλειδιού αποκρυπτογράφησης από το κλειδί κρυπτογράφησης. Αυτό έχει τεράστια πρακτική σημασία. Σε παραδοσιακά κρυπτογραφικά συστήματα, κάποιος μπορεί να στείλει κωδικοποιημένα μηνύματα μόνο αν και τα δύο άκρα γνωρίζουν το μυστικό κλειδί. Δεδομένου ότι όποιος μαθαίνει το κλειδί είναι σε θέση να αποκρυπτογραφήσει μηνύματα, τα κλειδιά πρέπει να φυλάσσονται προσεκτικά και να διαβιβάζονται μόνο κάτω από δρακόντεια

[1] Αυτό το πρόβλημα σχεδιάστηκε το 1987 από τους Ruth Shyu και Eric Grimson και αναθεωρήθηκε το 1992 από τους David Lamacchia και Hal Abelson.

[2] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, IT-22:6, 1976, pp 644–654.

μέτρα ασφαλείας. Στο σύστημα των Diffie και Hellman, μπορείτε να δώσετε το κλειδί *κρυπτογράφησης* σε οποιονδήποτε θέλει να σας στείλει μηνύματα, και να μην ανησυχείτε καθόλου για την ασφάλεια του κλειδιού. Γιατί, ακόμη και αν οποιοσδήποτε στον κόσμο ήξερε το κλειδί κρυπτογράφησης, κανείς δεν θα μπορούσε να αποκρυπτογραφήσει τα μηνύματα που σας αποστέλλονται χωρίς να γνωρίζει το *κλειδί αποκρυπτογράφησης*, το οποίο διατηρείτε ιδιωτικό. Οι Diffie και Hellman ονόμασαν ένα τέτοιο σύστημα: σύστημα κρυπτογράφησης *δημόσιου κλειδιού*.

Λίγους μήνες αφότου οι Diffie και Hellman ανακοίνωσαν την ιδέα τους, οι Ronald Rivest, Adi Shamir και Leonard Adelman από το MIT πρότειναν μια μέθοδο για την εφαρμογή της. Το *σύστημα κρυπτογράφησης RSA* έχει παραμείνει η πιο δημοφιλής τεχνική κρυπτογράφησης δημόσιου κλειδιού.

Η θεωρία πίσω από την κωδικοποίηση RSA

Η μέθοδος RSA χρησιμοποιεί ακέραιους αριθμούς για να εκπροσωπούν ομάδες χαρακτήρων [3] και χρησιμοποιεί ειδικές ρουτίνες που μετασχηματίζουν ακέραιους σε ακέραιους.

Στο RSA, επιλέγετε δύο μεγάλους πρώτους αριθμούς, p και q. Στη συνέχεια ορίζετε

$$n = pq \tag{1}$$
$$m = (p-1)(q-1). \tag{2}$$

Επίσης επιλέγετε έναν αριθμό e, έτσι ώστε $\gcd(e, m) = 1$. Το *δημόσιο κλειδί*, το οποίο διαμοιράζεστε με τον υπόλοιπο κόσμο, είναι το ζεύγος των αριθμών n και e. Όποιος θέλει να σας στείλει ένα μήνυμα s (το οποίο αναπαρίσταται ως ένας ακέραιος αριθμός) το κρυπτογραφεί χρησιμοποιώντας τον ακόλουθο μετασχηματισμό *RSA*, που ορίζεται από τα n και e:

κρυπτογραφημένο μήνυμα = s υψωμένο στην e, υπόλοιπο με n

ή

$$S = (s^e) \bmod n.$$

Αν λάβετε ένα κρυπτογραφημένο μήνυμα S, το αποκρυπτογραφείτε εκτελώντας έναν άλλο μετασχηματισμό RSA χρησιμοποιώντας το n και έναν ειδικό αριθμό d:

s' = κρυπτογραφημένο μήνυμα υψωμένο στην d, υπόλοιπο με n

ή

$$s' = (S^d) \bmod n.$$

Ο αριθμός d επιλέγεται έτσι ώστε να έχει την ιδιότητα $s = s'$ για κάθε μήνυμα s,[4] δηλαδή,

$$s = (s^e)^d \bmod n.$$

[3] Για παράδειγμα, το πρότυπο ASCII αναπαριστά κάθε χαρακτήρα με ακεραίους των 7-bit. Σε αυτό το πρόβλημα θα αντιπροσωπεύουμε ένα μπλοκ τεσσάρων χαρακτήρων ως ακεραίους των 28-bit ($0 \leq s < 2^{28}$) ενώνοντας τους κωδικούς ASCII του κάθε χαρακτήρα.

[4] Αυτό ισχύει αν-ν $\gcd(s, n) = 1$. Αν n είναι το γινόμενο δύο μεγάλων πρώτων αριθμών, τότε σχεδόν όλα τα μηνύματα $s < n$ ικανοποιούν την σχέση.

Αποδεικνύεται ότι ο αριθμός d που έχει αυτήν την ιδιότητα είναι αυτός για τον οποίον ισχύει

$$de = 1 \bmod m \tag{3}$$

δηλαδή, για κάποιο d που είναι *multiplicative inverse* του e υπόλοιπο με m.[5] Προκύπτει ότι είναι εύκολο να υπολογίσουμε αποδοτικά τον d, αν γνωρίζουμε τον e και τον $m = (p-1)(q-1)$.

Έτσι, για να δημιουργήσουμε ένα ζεύγος κλειδιών RSA, επιλέγουμε δύο πρώτους αριθμούς p και q, υπολογίζουμε το $n = pq$, επιλέγουμε ένα e, και χρησιμοποιούμε αυτό για να υπολογίσουμε το d. Μπορούμε να διαμοιράσουμε το ζευγάρι n και e ως το δημόσιο κλειδί μας, αλλά πρέπει να διατηρήσουμε το d κρυφό. Οι άλλοι μας στέλνουν κρυπτογραφημένα μηνύματα χρησιμοποιώντας το ζεύγος (n, e). Εμείς αποκρυπτογραφούμε τα μηνύματα χρησιμοποιώντας το ζεύγος (n, d).

Η ασφάλεια του συστήματος RSA βασίζεται στο γεγονός ότι, ακόμη και αν κάποιος γνωρίζει τα e και n, ο πιο αποτελεσματικός τρόπος για να αποκρυπτογραφήσουν ένα μήνυμα είναι να υπολογίσουν τους παράγοντες του n για να βρουν τα p και q, να τα χρησιμοποιήσουν για να υπολογίσουν το m, και να χρησιμοποιήσουν τα e και m για τον υπολογισμό του d.

Δηλαδή, το *σπάσιμο* ενός κωδικού RSA είναι, απ' όσον γνωρίζουμε, ένα υπολογιστικό πρόβλημα εξίσου δύσκολο με την εύρεση των πρώτων παραγόντων του n, p φορές q. Και παρόλο που υπήρξε εκτενής έρευνα για μεθόδους παραγοντοποίησης, η εφαρμογή τους σε αυθαίρετα μεγάλους αριθμούς δεν είναι υπολογιστικά εφικτή. Για παράδειγμα, η παραγοντοποίηση του $n = pq$ όπου p και q είναι έκαστος πρώτος αριθμός των 200 ψηφίων, ακόμη και με τον καλύτερο αλγόριθμο παραγοντοποίησης, απαιτεί υπολογισμούς για χρόνια στους ταχύτερους υπολογιστές. [6]

Ψηφιακές υπογραφές: Κωδικοποίηση και υπογραφή

Στο άρθρο τους το 1976, οι Diffie και Hellman πρότειναν την εφαρμογή της κρυπτογράφησης δημόσιου κλειδιού για την επίλυση ενός άλλου σημαντικού προβλήματος της ασφαλούς επικοινωνίας. Το πρόβλημα είναι το εξής: ας υποθέσουμε ότι θέλετε να στείλετε ένα μήνυμα μέσω ηλεκτρονικού ταχυδρομείου. Πώς μπορούν οι άνθρωποι που λαμβάνουν το μήνυμα να είναι βέβαιοι ότι προέρχεται πραγματικά από εσάς — ότι δεν είναι πλαστό; Απαιτείται κάποιο σύστημα για την επισήμανση ενός μηνύματος με τέτοιο τρόπο ώστε να μην μπορεί να παραποιηθεί. Μία τέτοια μέθοδος είναι η *ψηφιακή υπογραφή*.

Η πρόταση των Diffie και Hellman ήταν η εξής: παίρνουμε το μήνυμα και εφαρμόζουμε μια εκ των προτέρων συμφωνημένη *διεργασία συμπίεσης* (ονομάζεται επίσης *συνάρτηση κατακερματισμού*) που μετατρέπει όλο το μήνυμα σε ένα τελικό, σχετικά μικρό αριθμό. Γενικά, θα υπάρχουν πολλά μηνύματα που παράγουν την ίδια τιμή κατακερματισμού. Τώρα μετασχηματίζουμε αυτή την τιμή χρησιμοποιώντας το ιδιωτικό μας κλειδί. Η νέα τιμή κατακερματισμού είναι η ψηφιακή μας υπογραφή, η οποία θα διαβιβαστεί μαζί με το μήνυμα. Οποιοσδήποτε λαμβάνει ένα μήνυμα μπορεί να πιστοποιήσει την υπογραφή, μετατρέποντας την χρησιμοποιώντας το *δημόσιο κλειδί* μας και ελέγχει ότι το αποτέλεσμα είναι το ίδιο με την τιμή της συνάρτησης κατακερματισμού εφαρμοσμένη στο μήνυμα.

[5] Αυτό είναι ένα βασικό αποτέλεσμα της θεωρίας αριθμών, εμείς απλά θα σας ζητήσουμε να το πιστέψετε.

[6] Κανείς δεν έχει στην πραγματικότητα αποδείξει ότι το σπάσιμο ενός κωδικού RSA είναι εξίσου δύσκολο πρόβλημα όπως η παραγοντοποίηση, αλλά δεν έχει βρεθεί άλλη μέθοδος. Επιπλέον, ορισμένοι επιστήμονες πιστεύουν ότι ίσως είναι δυνατό να αποδειχθεί ότι δεν μπορεί να υπάρξει γρήγορος (π.χ., λογαριθμικής πολυπλοκότητας) αλγόριθμος για παραγοντοποίηση. Δεδομένης της δημοτικότητας της RSA, η ανακάλυψη ενός τέτοιου αλγορίθμου θα οδηγήσει σε μαζικές παραβιάσεις ασφάλειας για τράπεζες, επιχειρήσεις και άλλους οργανισμούς που χρησιμοποιούν RSA.

Ο λόγος που αυτό το σύστημα λειτουργεί είναι διότι, αν κάποιος θέλει να πλαστογραφήσει ένα μήνυμα, ισχυριζόμενος ότι είναι από εσάς, πρέπει να παράξει έναν αριθμό που, όταν μεταμορφωθεί από το δημόσιο κλειδί σας, να ταιριάζει με την τιμή κατακερματισμού. Οποιοσδήποτε μπορεί να υπολογίσει την τιμή κατακερματισμού, εφόσον η συνάρτηση είναι δημόσια. Αλλά από τη στιγμή που είστε οι μόνοι που γνωρίζετε το ιδιωτικό σας κλειδί, μόνο εσείς μπορείτε να παράξετε τον αριθμό που τελικά μετασχηματίζεται στην τιμή κατακερματισμού από το δημόσιο κλειδί σας. Η παραποίηση της ψηφιακής υπογραφής είναι ουσιαστικά το ίδιο δύσκολο με το σπάσιμο ενός κρυπτογραφημένου μηνύματος.

Μια ακόμη καλύτερη ιδέα είναι η ακόλουθη: Ας υποθέσουμε ότι η Barbara θέλει να στείλει στον George ένα υπογεγραμμένο μήνυμα που μόνο αυτός θα είναι σε θέση να διαβάσει. Αυτή κρυπτογραφεί το μήνυμα χρησιμοποιώντας το δημόσιο κλειδί του Γιώργου. Στη συνέχεια υπογράφει το κρυπτογραφημένο αποτέλεσμα χρησιμοποιώντας το δικό της ιδιωτικό κλειδί. Όταν ο Γιώργος λάβει ένα μήνυμα που υποτίθεται ότι είναι από την Barbara, χρησιμοποιεί αρχικά το δημόσιο κλειδί της για την επικύρωση της υπογραφής και, στη συνέχεια, αποκρυπτογραφεί το μήνυμα χρησιμοποιώντας το ιδιωτικό του κλειδί. Το Σχήμα 1 απεικονίζει την μέθοδο.

Παρατηρήστε τι επιτυγχάνουμε έτσι: Ο George μπορεί να είναι βέβαιος ότι μόνο κάποιος με το ιδιωτικό κλειδί της Barbara θα μπορούσε να του στείλει το μήνυμα. Η Barbara μπορεί να είναι σίγουρη ότι μόνο κάποιος με το ιδιωτικό κλειδί του Γιώργου μπορεί να διαβάσει το μήνυμα. Αυτό επιτυγχάνεται χωρίς την ανταλλαγή οποιασδήποτε μυστικής πληροφορίας μεταξύ του George και της Barbara. Αυτή η δυνατότητα επίτευξης ασφαλούς επικοινωνίας χωρίς να απαιτείται η ανταλλαγή μυστικών κλειδιών καθιστά την κρυπτογραφία δημόσιου κλειδιού μία ιδιαιτέρως σημαντική τεχνική.

Υλοποιώντας τον RSA

Το βασικό στοιχείο που χρειαζόμαστε για την υλοποίηση της RSA είναι η μέθοδος fast exponentiation, όπως περιγράφηκε στην παράγραφο 1.2.6 του βιβλίου:

```
(define (expmod b e m)
  (cond ((zero? e) 1)
        ((even? e)
         (remainder (square (expmod b (/ e 2) m)) m))
        (else (remainder (* b (expmod b (- e 1) m)) m))))
```

Υποθέτουμε ότι ένα κλειδί RSA απεικονίζεται ως ένα ζεύγος (διαιρέτης και εκθέτης):

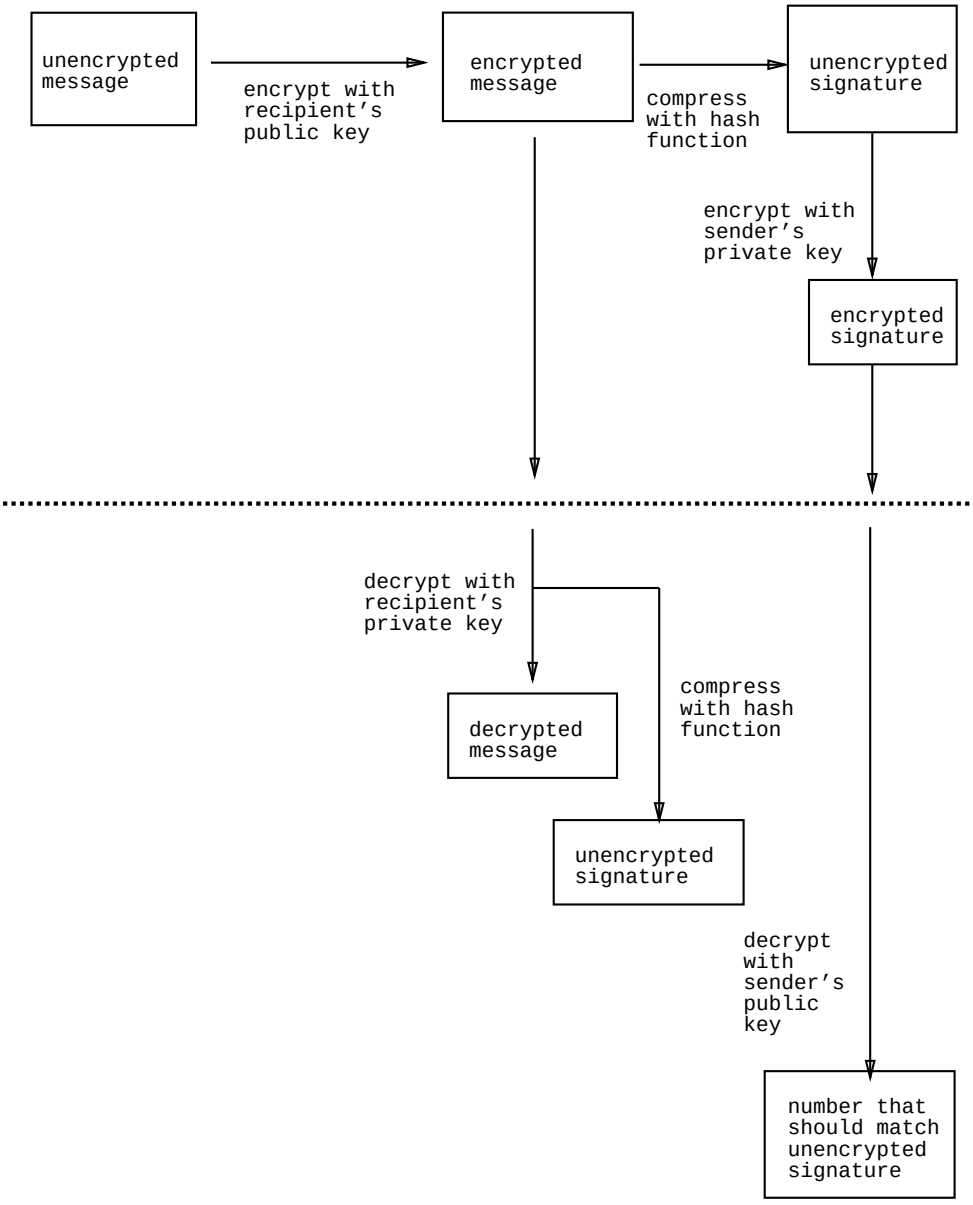
```
(define make-key cons)
(define key-modulus car)
(define key-exponent cdr)
```

Ο βασικός μετασχηματισμός RSA τότε είναι

```
(define (RSA-transform number key)
  (expmod number (key-exponent key) (key-modulus key)))
```

Παραγωγή πρώτων αριθμών

Για την δημιουργία κλειδιών RSA, πρώτα απ' όλα χρειάζεται ένας τρόπος δημιουργίας πρώτων αριθμών. Ο πιο απλός τρόπος είναι να διαλέξετε ένα τυχαίο αριθμό σε μία επιθυμητή περιοχή και να κάνετε δοκιμές σε



Σχήμα 1: Κρυπτογράφηση και ψηφιακή υπογραφή.

διαδοχικούς αριθμούς μέχρι να βρείτε έναν πρώτο. Η ακόλουθη διαδικασία ξεκινά την αναζήτηση σε μια τυχαία περιοχή ανάμεσα στους ακέραιους `start` και `start + range`:

```
(define (choose-prime smallest range)
  (let ((start (+ smallest (choose-random range))))
    (search-for-prime (if (even? start) (+ start 1) start))))

(define (search-for-prime guess)
  (if (fast-prime? guess 2)
      guess
      (search-for-prime (+ guess 2))))

(define choose-random
  ;; restriction of Scheme RANDOM primitive
  (let ((max-random-number (expt 10 18)))
    (lambda (n)
      (random (floor->exact (min n max-random-number))))))
```

Για τον έλεγχο αν ένας αριθμός είναι πρώτος χρησιμοποιούμε το τεστ του Fermat, όπως περιγράφεται στην παράγραφο 1.2.6:

```
(define (fermat-test n)
    (let ((a (choose-random n)))
      (= (expmod a n n) a)))

(define (fast-prime? n times)
    (cond ((zero? times) true)
          ((fermat-test n) (fast-prime? n (- times 1)))
          (else false)))
```

Παραγωγή ζεύγους κλειδιών RSA

Τώρα μπορούμε να δημιουργήσουμε ένα δημόσιο κλειδί RSA και το αντίστοιχο ιδιωτικό. Αναπαριστούμε αυτό το ζεύγος ως εξής:

```
(define make-key-pair cons)
(define key-pair-public car)
(define key-pair-private cdr)
```

Η ακόλουθη διαδικασία δημιουργεί ένα ζεύγος κλειδιών RSA. Επιλέγει τους πρώτους P και Q που βρίσκονται στην περιοχή από 2^{14} έως 2^{15}, έτσι ώστε ο ακέραιος $n = pq$ να ανήκει στο εύρος 2^{28} με 2^{30}, το οποίο είναι αρκετά μεγάλο για την κωδικοποίηση τεσσάρων χαρακτήρων ανά αριθμό. [7] Μετά την επιλογή των πρώτων αριθμών, υπολογίζει τους n και m σύμφωνα με τις εξισώσεις (1) και (2). Στη συνέχεια επιλέγει έναν εκθέτη e και βρίσκει έναν αριθμό d που ικανοποιεί την εξίσωση (3).

[7] Χρησιμοποιούμε τέτοιες μικρές τιμές του n για αυτό το πρόβλημα, επειδή θέλουμε να πειραματιστείτε με το σπάσιμο ενός συστήματος RSA. Ξεκινώντας με μεγαλύτερους τυχαίους αριθμούς, μπορείτε να χρησιμοποιήσετε την ίδια μέθοδο για να παράξετε πραγματικά ασφαλές σύστημα.

```
(define (generate-RSA-key-pair)
  (let ((size (expt 2 14)))
    (let ((p (choose-prime size size))
          (q (choose-prime size size)))
      (if (= p q)                 ;check that we haven't chosen the same prime twice
          (generate-RSA-key-pair) ;(VERY unlikely)
          (let ((n (* p q))
                (m (* (- p 1) (- q 1))))
            (let ((e (select-exponent m)))
              (let ((d (invert-modulo e m)))
                (make-key-pair (make-key n e) (make-key n d)))))))))
```

Ο εκθέτης e μπορεί να είναι οποιοσδήποτε τυχαίος αριθμός $0 < e < m$ με $\gcd(e, m) = 1$. Η διαδικασία `gcd` δίνεται στην παράγραφο 1.2.5 των σημειώσεων, αλλά είναι στην πραγματικότητα primitive ρουτίνα της Scheme.

```
(define (select-exponent m)
  (let ((try (choose-random m)))
    (if (= (gcd try m) 1)       ;if gcd is not 1, then try again
        try
        (select-exponent m))))
```

Υπολογισμός του multiplicative inverse

Ο αριθμός d που απαιτείται για το κλειδί RSA πρέπει να ικανοποιεί την εξίσωση

$$de = 1 \bmod m$$

Χρησιμοποιώντας τον ορισμό του equality modulo m, αυτό σημαίνει ότι το d πρέπει να ικανοποιεί

$$km + de = 1$$

όπου k ένας (αρνητικός) ακέραιος. Αποδεικνύεται ότι λύση στην εξίσωση αυτή υπάρχει αν-ν $\gcd(e, m) = 1$. Η ακόλουθη διαδικασία παράγει την απαιτούμενη τιμή d, υποθέτοντας ότι έχουμε διαθέσιμη μία ρουτίνα η οποία, δοθέντος δύο ακεραίων a και b, επιστρέφει ένα ζευγάρι ακεραίων (x, y) έτσι ώστε $ax + by = 1$. [8]

```
(define (invert-modulo e m)
  (if (= (gcd e m) 1)
      (let ((y (cdr (solve-ax+by=1 m e))))
        (modulo y m))   ;take y modulo m, in case y was negative
      (error "gcd not 1" e m)))
```

Η επίλυση της $ax + by = 1$ μπορεί να επιτευχθεί με ένα αναδρομικό τέχνασμα που σχετίζεται με το αναδρομικό αλγόριθμο GCD , όπως περιγράφεται στο τμήμα 1.2.5 του βιβλίου. Αν q είναι το πηλίκο του a με b, και r είναι το υπόλοιπο του a με b, έτσι ώστε

$$a = bq + r$$

η αναδρομική επίλυση της εξίσωσης

$$b\bar{x} + r\bar{y} = 1$$

μπορεί να χρησιμοποιηθεί ώστε, μέσω των $\bar{x}$ και $\bar{y}$ να παραχθούν τα x και y. Θα χρειαστεί να υλοποιήσετε εσείς αυτή τη διαδικασία.

[8] Η primitive ρουτίνα της Scheme `modulo`, την οποία χρησιμοποιούμε για να αναγκάσουμε θετικό αποτέλεσμα, είναι η ίδια με την `remainder`, εκτός από αρνητικά ορίσματα: `(remainder -12 7)` δίνει -5, ενώ το `(modulo -12 7)` δίνει 2. Γενικά, το `(modulo a b)` έχει πάντα το ίδιο πρόσημο με το `b`, ενώ το `(remainder a b)` έχει πάντα το ίδιο πρόσημο με το `a`.

Κωδικοποίηση - Αποκωδικοποίηση μηνυμάτων

Τέλος, για την χρήση RSA, χρειαζόμαστε έναν τρόπο μετατροπής μεταξύ string και αριθμών. Ο κώδικας του tester3.scm περιλαμβάνει τις ρουτίνες `string->intlist` και `intlist->string` οι οποίες εκτελούν τις αντίστοιχες μετατροπές (από string σε λίστα ακεραίων και αντίστροφα). Κάθε ακέραιος (μεταξύ 0 και 2^{28}) κωδικοποιεί 4 διαδοχικούς χαρακτήρες του μηνύματος. Εάν ο αριθμός των χαρακτήρων δεν είναι πολλαπλάσιο του 4, τότε παρατίθενται κενοί χαρακτήρες στο τέλος του:

```
(string->intlist "This is a string.")
;Value: (242906196 69006496 245157985 217822450 67637294)

(intlist->string '(242906196 69006496 245157985 217822450 67637294))
;Value: "This is a string.   "
```

Η υλοποίηση αυτών των διαδικασιών δεν σας απασχολεί. Μπορείτε να τις μελετήσετε αν θέλετε να μάθετε πως χειρίζεται συμβολοσειρές (strings) η Scheme.

Για να κρυπτογραφήσουμε ένα μήνυμα, μετατρέπουμε το μήνυμα σε μια λίστα ακεραίων και μετασχηματίζουμε τη λίστα χρησιμοποιώντας τη μέθοδο RSA μαζί με το ένα κλειδί από ένα ζεύγος κλειδιών.

```
(define (RSA-encrypt string key1)
  (RSA-convert-list (string->intlist string) key1))
```

Ίσως να μαντέψατε ότι ο σωστός τρόπος για την κωδικοποίηση της λίστας των αριθμών θα είναι η ξεχωριστή κωδικοποίηση κάθε αριθμού. Αλλά αυτό δεν λειτουργεί ικανοποιητικά. Αντ' αυτού, κρυπτογραφούμε τον πρώτο αριθμό, τον αφαιρούμε από το δεύτερο αριθμό (modulo n) και κρυπτογραφούμε το αποτέλεσμα, και ούτω καθεξής, έτσι ώστε κάθε αριθμός στην τελική λίστα να εξαρτάται όλους τους προηγούμενους:

```
(define (RSA-convert-list intlist key)
  (let ((n (key-modulus key)))
    (define (convert l sum)
      (if (null? l)
          '()
          (let ((x (RSA-transform (modulo (- (car l) sum) n)
                                  key)))
            (cons x (convert (cdr l) x)))))
    (convert intlist 0)))
```

Θα αφήσουμε σε σας την υλοποίηση της ανάλογης διαδικασίας (`RS-unconvert-list`) που αναιρεί αυτό το μετασχηματισμό χρησιμοποιώντας το άλλο κλειδί του ζεύγος. Έπειτα εκτελούμε

```
(define (RSA-decrypt intlist key2)
  (intlist->string (RSA-unconvert-list intlist key2)))
```

Τέλος, για τη δημιουργία των ψηφιακών υπογραφών σε κρυπτογραφημένα μηνύματα, χρειαζόμαστε μια τυπική λειτουργία συμπίεσης. Στην παρούσα υλοποίηση, απλά θα προσθέσουμε τους ακέραιους modulo 2^{28}. [9]

```
(define (compress intlist)
  (define (add-loop l)
    (if (null? l)
        0
        (+ (car l) (add-loop (cdr l)))))
  (modulo (add-loop intlist) (expt 2 28)))
```

[9] Στην πράξη, χρησιμοποιούνται πιο περίπλοκα συστήματα συμπίεσης. Μπορείτε να σκεφτείτε γιατί;

3. Υπόβαθρο για την εργασία

Ministry of Information

To: Ross (the Boss)
From: Rupert

So far we've been pretty successful. I really liked the way you arranged that cattle-futures deal, and the creative accounting by our mole in the Rose Law firm has really done wonders. But I'm getting concerned about the security of our network. My $4M book deal with the Salamander got out before the optimal moment. I hope we haven't been cracked by the entity in Fort Meade.

Central Control

To: Rupert
From: Ross

You're absolutely right about the need for security. I've gotten in touch with some people I know at Family Values Communications. FVC markets a system that encrypts and authenticates messages using a technique called RSA. The FVC people say they can build an encryption system for the modest fee of $120M.

Ministry of Information

To: Ross
From: Rupert

$120 million?!? *You have to be kidding.* That's almost as much as it cost us to replace Gorby with Boris. I contacted Chuck (the Vest) at New England Research and Development (His cover is President of MIT.) to ask his advice. As you know, he helped us arrange the White House mail system.[10] Chuck says he can do the job for us, for a minor consideration. He needs help getting John (the German) installed in the entity in Virginia.

MASSACHVSETTS INSTITUTE OF TECHNOLOGY
Office of the President

Dear Albert and Gerry:

I have received a request of the *highest priority* asking that 6.001's next problem set involve RSA cryptography and digital signatures. Sorry for the rush. I've managed to get some of the code from Family Values Communications, so at least the students won't be starting from scratch. Thanks!

Chuck Vest

---

[10] This is really true. The electronic mail connection to the White House was set up by people at the MIT AI Lab.

4. Άσκηση προγραμματισμού!

Αρχικά φορτώστε τον κώδικα και ελέγξτε ότι τρέχει η tester3.scm.

Ελέγξτε την σωστή χρήση των παρακάτω

```
(define test-public-key1 (key-pair-public test-key-pair1))
(define result1 (rsa-encrypt "This is a test message." test-public-key1))
```

Το `Result1` πρέπει να ισούται με

```
(209185193 793765302 124842465 169313344 117194397 237972864)
```

Το ζεύγος `test-key-pair1` είναι ένα πρότυπο ζεύγος κλειδιών RSA που δημιουργήθηκαν για να δοκιμάσετε τον κώδικα. Προσοχή, καθώς η στίξη και τα κεφαλαία επηρεάζουν το αποτέλεσμα.

Άσκηση 1: Δυστυχώς, ο κώδικας που δίνεται από τον πρόεδρο είναι ελλιπής ως προς μία συνάρτηση (`RSA-unconvert-list`) η οποία είναι απαραίτητη για την αποκρυπτογράφηση μηνυμάτων.

Υλοποιήστε την μέθοδο η οποία παίρνει ως όρισμα μία λίστα από ακεραίους προς αποκρυπτογράφηση και ένα κλειδί αποκρυπτογράφησης, και επιστρέφει μία λίστα ακεραίων, αντιστρέφοντας τον μετασχηματισμό της `RSA-convert-list`. Hint: Η διαδικασία θα μοιάζει πολύ με την `RSA-convert-list`.

Δοκιμάστε την,

```
(define test-private-key1 (key-pair-private test-key-pair1))

(RSA-unconvert-list result1 test-private-key1)
```

Το αποτέλεσμα πρέπει να είναι

```
(242906196 69006496 213717089 229128819 205322725 67875559)
```

Αν αυτό λειτουργεί τότε η παρακάτω εντολή

```
(RSA-decrypt result1 test-private-key1)
```

θα επιστρέψει το αρχικό αποκρυπτογραφημένο μήνυμα (εκτός από κάποια κενά στο τέλος).

Ένα δεύτερο ζεύγος κλειδιών για δοκιμές μπορεί να ληφθεί εκτελώντας:

```
(define test-public-key2 (key-pair-public test-key-pair2))
(define test-private-key2 (key-pair-private test-key-pair2))
```

Άσκηση 2: Υλοποιήστε τη μέθοδο κρυπτογράφησης και υπογραφής ενός μηνύματος, όπως περιγράφεται στην ενότητα 1.

Αρχικά, ορίστε μία απλή δομή δεδομένων ονόματι `signed-message` που αποτελείται από ένα μήνυμα `message` και από μία υπογραφή `signature`. Στη συνέχεια, ορίστε τη διαδικασία `encrypt-and-sign` η οποία παίρνει ως όρισμα το μήνυμα, το ιδιωτικό κλειδί του αποστολέα και το δημόσιο κλειδί του δέκτη. Η διαδικασία θα κρυπτογραφεί το μήνυμα, θα παράγει την ψηφιακή υπογραφή του, και θα τα ενώνει για να δημιουργήσει το signed message.

Για δοκιμή,

```
(define result2
  (encrypt-and-sign "Test message from user 1 to user 2"
                    test-private-key1
                    test-public-key2))
```

Το αποτέλεσμα θα πρέπει να είναι ένα signed-message με μήνυμα

```
(499609777 242153055 12244841 376031918 242988502 31156692 221535122 463709109 468341391)
```

και υπογραφή 15378444.

Τώρα υλοποιήστε την αντίστροφη διαδικασία `authenticate-and-decrypt`, η οποία παίρνει σαν όρισμα το κρυπτογραφημένο μήνυμα, το δημόσιο κλειδί του αποστολέα, και το ιδιωτικό κλειδί του δέκτη. Αν η υπογραφή είναι αυθεντική, αποκρυπτογραφείται το μήνυμα. Αλλιώς, επιστρέφεται η ένδειξη ”Authentication failed!”.

Για δοκιμή,

```
(authenticate-and-decrypt result2 test-public-key1 test-private-key2)
```

για να ανακτήσετε το αρχικό μήνυμα.

Άσκηση 3: Το δημόσιο κλειδί του Bill Clinton ορίζεται στην ps3.scm:

```
(define bill-clinton-public-key (make-key 833653283 583595407))
```

Επίσης ορίζονται τα ακόλουθα δημόσια κλειδιά:

```
(define al-gore-public-key (make-key 655587853 463279441))
(define bob-dole-public-key (make-key 507803083 445001911))
(define ross-perot-public-key (make-key 865784123 362279729))
(define hillary-clinton-public-key (make-key 725123713 150990017))
(define tipper-gore-public-key (make-key 376496027 270523157))
(define chuck-vest-public-key (make-key 780450379 512015071))
(define rupert-murdoch-public-key (make-key 412581307 251545759))
(define newt-gingrich-public-key (make-key 718616329 290820109))
```

Χθες ο Gingrich έλαβε το ακόλουθο μήνυμα:

```
(510560918 588076790 115222453 249656722 408910590 69814552
 690687967 281490047 41430131 256420885 184791295 75938032
 693840839 663727111 593617709 335351412)
```

Η υπογραφή ήταν 65732336. (Οι τιμές είναι ορισμένες στην tester.scm ως `received-mystery-message` και `received-mystery-signature`.) Ευτυχώς για μας, ένα φίλος κατάφερε να αποκτήσει το ιδιωτικό κλειδί του Gingrich:

```
(define newt-gingrich-private-key (make-key 718616329 129033029))
```

Αποκρυπτογραφήστε το μήνυμα και βρείτε ποιος το έστειλε. Υλοποιήστε την συνάρτηση

```
(define (decrypt-and-identify) '<YOUR-CODE-HERE>)
```

η οποία επιστρέφει το αποκρυπτογραφημένο μήνυμα σε μορφή `string`.

Άσκηση 4: Οι φίλοι μας στο FVC μας έστειλαν επίσης μία συνάρτηση που παράγει ένα ζεύγος κλειδιών RSA: το δημόσιο και το συσχετιζόμενο ιδιωτικό. Τους λείπει, όμως, η διεργασία που λύνει εξισώσεις της μορφής $ax + by = 1$. Υλοποιήστε την, με όνομα `solve-ax+by=1`. Δέχεται ως είσοδο δύο ορίσματα, τους

ακέραιους a και b των οποίων ο μέγιστος κοινός διαιρέτης (GCD) είναι ίσος με 1. Επιστρέφει ένα ζεύγος ακεραίων x and y. Ελέγξτε την ορθότητα της επιλύοντας την παρακάτω εξίσωση:

$$233987973x + 41111687y = 1$$

Αν υλοποιήσατε σωστά την προηγούμενη συνάρτηση, μπορείτε πλέον να καλέσετε την `generatersa-key-pair` (χωρίς κανένα όρισμα) για να δημιουργήσετε ένα τυχαίο ζεύγος κλειδιών. Δημιουργήστε ένα για τον εαυτό σας.

Άσκηση 5: Έχετε πλέον υλοποιημένο ένα πλήρες σύστημα κρυπτογράφησης RSA, με όλες τις απαραίτητες ρουτίνες. Εφόσον η υλοποίηση αυτή χρησιμοποιεί μικρούς πρώτους αριθμούς, μπορείτε να *σπάσετε* το σύστημα. Για να το καταφέρετε, θυμηθείτε ότι πρέπει να βρείτε δύο πρώτους παράγοντες p and q για το n. Χρησιμοποιήστε την `smallest-divisor` που σας παρέχεται.[11] Υλοποιήστε την ρουτίνα `crack-rsa` η οποία, δοθέντος ενός δημόσιου κλειδιού, επιστρέφει το ιδιωτικό. Ελέγξτε την χρησιμοποιώντας τα δύο ζεύγη `test-key-pair1` και `test-key-pair2` που σας δίνονται.

Άσκηση 6: Ο Bob Dole επιθυμεί να κοροϊδέψει την διοίκηση Clinton ώστε να πάρει σκληρά μέτρα. Παραποιήστε το μήνυμα που είναι αποθηκευμένο στη μεταβλητή `clinton-to-gore-fake` από τον Clinton στον Gore, ζητώντας του να ανακοινώσει ότι ετοιμάζουν αύξηση φόρων. Η συνάρτηση

```
(define (forge-message message s-public-key r-public-key))
```

θα δέχεται ως είσοδο ένα κείμενο (`message`), τα δημόσια κλειδιά του αποστολέα (`s-public-key`) και του παραλήπτη (`r-public-key`) και θα επιστρέφει ένα κωδικοποιημένο μήνυμα τύπου `signed-message`.

[11] Όταν βρείτε έναν πρώτο παράγοντα p, ο άλλος είναι $q = n/p$:-).