Αλγόριθμοι Πιστοποίησης Πρώτων Αριθμών σε Ενσωματωμένα Συστήματα

Αλεξάνδρα Τσιφόρα
Τμήμα Μαθηματικών
Πανεπιστήμιο Αιγαίου

Ιούνιος 2009

Επιβλέπων: Παναγιώτης Νάστου

Τριμελής Επιτροπή
Ευάγγελος Φελουζής, Επίκουρος
Παναγιώτης Νάστου, Λέκτορας
Ιωάννης Σταματίου, Επίκουρος

Ευχαριστώ πολύ τον επιβλέποντα καθηγητή μου Παναγιώτη Νάστου για την καθοδήγηση, την υπομονή και τον χρόνο που αφιέρωσε για την επίτευξη της παρούσας μεταπτυχιακής εργασίας.

Επίσης ευχαριστώ την οικογένεια μου και τους φίλους μου Βαγγέλη Στάθη, Γιάννη Μάνεση και Μαγδαληνή Πλιόγκα για την ψυχολογική υποστήριξη.

Αφιερωμένη στους:

Μάνο, Πάρη, Μαρία, Χρήστο, Ζαφείρη, Στέλιο

Α. Τσιφόρα, Σάμος 2009.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Η λέξη κρυπτογραφία προέρχεται από τα συνθετικά κρύπτω και γράφω και είναι ένας επιστημονικός κλάδος που ασχολείται με τη μελέτη, την ανάπτυξη και τη χρήση τεχνικών κρυπτογράφησης και αποκρυπτογράφησης με σκοπό την απόκρυψη του περιεχομένου των μηνυμάτων. Η κρυπτογραφία είναι ένας κλάδος της επιστήμης της κρυπτολογίας η οποία ασχολείται με τη μελέτη της ασφαλούς επικοινωνίας. Ο κύριος στόχος της είναι να παρέχει μηχανισμούς για 2 ή περισσότερα μέλη να επικοινωνήσουν χωρίς κάποιος άλλος να διαβάζει την πληροφορία εκτός από τα μέλη.

Στην κρυπτογραφία διακρίνονται 4 βασικές υπηρεσίες:

1. **Εμπιστευτικότητα**: Αφορά την προστασία από τη μη εξουσιοδοτημένη αποκάλυψη της πληροφορίας. Η εμπιστευτικότητα θα πρέπει να προσφέρεται με τέτοιον τρόπο ώστε να είναι αδύνατη η αποκάλυψη και πολλές φορές η ίδια η ύπαρξη της πληροφορίας σε μη εξουσιοδοτημένα άτομα.

2. **Ακεραιότητα**: Είναι η προστασία από το μη εξουσιοδοτημένο παραλήπτη των δεδομένων. Η ακεραιότητα θα πρέπει να παρέχει στον παραλήπτη και γενικότερα στον κάτοχο ενός μηνύματος τη δυνατότητα να μπορεί να ανιχνεύσει πιθανές αλλαγές στο μήνυμα από μη εξουσιοδοτημένα άτομα. Στο χώρο των τηλεπικοινωνιών και της θεωρίας της πληροφορίας η ακεραιότητα είναι γνωστή ως ανίχνευση σφαλμάτων, όπου ένα μήνυμα μπορεί να υποστεί τροποποίηση λόγω του θορύβου του καναλιού επικοινωνίας.

3. **Μη απάρνηση**: Είναι η υπηρεσία κατά την οποία ο παραλήπτης δεν μπορεί να απαρνηθεί ότι έλαβε το μήνυμα (μη απάρνηση προορισμού-non-repudation of destination), ή η υπηρεσία κατά την οποία ο αποστολέας δεν μπορεί να απαρνηθεί ότι έστειλε το μήνυμα (μη απάρνηση προέλευσης -non-repudiation of origin).

4. **Αυθεντικοποίηση**: Είναι η εξασφάλιση του ότι γνωρίζουμε την οντότητα που επικοινωνούμε (user/entity authentication). Αυθεντικοποίηση δεδομένων (data authentication) είναι η εξασφάλιση ότι ένα μήνυμα προέρχεται πράγματι από τον αποστολέα που πιστεύουμε ότι το έστειλε.

1.1 Κατηγορίες Κρυπτοσυστημάτων

Τα κρυπτοσυστήματα διακρίνονται στα κλασσικά και στα μοντέρνα. Στα μοντέρνα κρυπτοσυστήματα εντάσσονται τα συμμετρικά και τα ασύμμετρα. Στο παρόν εδάφιο θα παρουσιασθούν οι βασικές αρχές λειτουργίας τόσο των συμμετρικών όσο και των ασύμμετρων κρυπτοσυστημάτων.

1.1.1 Συμμετρικά κρυπτοσυστήματα

Η συμμετρική κρυπτογραφία είναι κατά πολύ αρχαιότερη από την ασύμμετρη κρυπτογραφία. Χρονολογείται από την Αρχαία Αίγυπτο, ενώ η ασύμμετρη κρυπτογραφία εμφανίστηκε τη δεκαετία του ′70. Στη συμμετρική κρυπτογράφηση χρησιμοποιείται το ίδιο κλειδί για την κρυπτογράφηση και την αποκρυπτογράφηση όπως φαίνεται και στο Σχήμα 1.1. Το κλειδί αυτό θα πρέπει να είναι γνωστό μόνο στα εξουσιοδοτημένα μέλη, και κατά συνέπεια, απαιτείται κάποιο ασφαλές μέσο για τη μετάδοσή του, όπως μια προσωπική συνάντηση, κατά την οποία θα συμφωνηθεί το κλειδί που θα χρησιμοποιείται. Αν κάτι τέτοιο δεν είναι εφικτό, η συμμετρική κρυπτογράφηση είναι αναποτελεσματική.

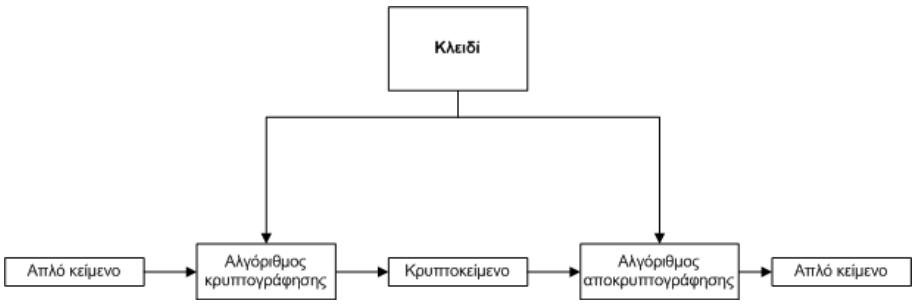
Τα συστήματα συμμετρικής κρυπτογράφησης προυποθέτουν την ύπαρξη ενός ασφαλούς καναλιού για την ανταλλαγή των μυστικών κλειδιών. Τέτοια συστήματα έχουν αναπτυχθεί και ήδη χρησιμοποιούνται, με πιο διαδεδομένο το σύστημα Kerberos, του MIT (Massachusetts Institute of Technology).

Οι συμμετρικοί κρυπταλγόριθμοι μπορούν να χωριστούν σε δύο διαφορετικές κατηγορίες με βάση τον τρόπο κρυπτογράφησης των μηνυμάτων:

- Τμήματος: οι οποίοι χωρίζουν το μήνυμα σε κομμάτια και κρυπτογραφούν κάθε ένα από τα κομμάτια αυτά χωριστά.
- Ροής: οι οποίοι κρυπτογραφούν μια ροή μηνύματος χωρίς να τη διαχωρίζουν σε τμήματα.

Κρυπταλγόριθμοι τμήματος

Σήμερα, στην πλειοψηφία τους τα συμμετρικά κρυπτοσυστήματα βασίζονται σε κρυπταλγόριθμους τμήματος. Η σχετικά μεγαλύτερη ασφάλεια που παρέχουν



Σχήμα 1.1: Αρχή λειτουργίας συμμετρικού κρυπτοσυστήματος

οι κρυπταλγόριθμοι τμήματος έναντι των κρυπταλγόριθμων ροής καθιστούν τους κρυπταλγόριθμους τμήματος ως πρώτη επιλογή.

Τακτικές σχεδιασμού: Ένας κρυπταλγόριθμος τμήματος είναι συνήθως μια επαναληπτική εφαρμογή μιας κρυπτογραφικής πράξης η οποία αποτελείται από μία ή περισσότερες κρυπτογραφικές συναρτήσεις, σε διάταξη τέτοια ώστε να επιτρέπεται η διαδοχική σύνδεση της πράξης αυτής με τον εαυτό της ή με διαφορετικές πράξεις. Το αποτέλεσμα της πράξης αυτής αποτελεί το κρυπτογραφικό γινόμενο, το οποίο υπό κατάλληλες συνθήκες μπορεί να καταστήσει ένα κρυπτοσύστημα ασφαλές. Το κάθε συστατικό του κρυπτογραφικού γινομένου αποτελεί το γύρο του κρυπταλγόριθμου. Σε κάθε γύρο τροφοδοτείται το αποτέλεσμα του προηγούμενου γύρου και το αντίστοιχο κλειδί.

Η ακολουθία των κλειδιών όλων των γύρων αποτελεί το πρόγραμμα κλειδιού και προκύπτει από κάποιο αρχικό κλειδί. Στον πρώτο γύρο τροφοδοτείται το απλό κείμενο, ενώ το αποτέλεσμα του τελευταίου γύρου αποτελεί το κρυπτοκείμενο. Ο αριθμός των γύρων εξαρτάται από την κρυπτογραφική δύναμη του κάθε γύρου. Γενικά το κρυπτογραφικό γινόμενο δύο σχετικά αδύναμων κρυπτογραφικών πράξεων ισοδυναμεί σε μια κρυπτογραφική πράξη η οποία είναι κατά πολύ δυνατότερη κρυπτογραφικά από τις επιμέρους πράξεις. Αυτό σύμφωνα με τον Shannon ονομάζεται ¨φαινόμενο της χιονοστοιβάδας (avalanche effect)¨.

Τα πιο γνωστά συμμετρικά κρυπτοσυστήματα τμήματος είναι:

- ο κρυπταλγόριθμος DES (Data Encryption Standard).

  Ο DES δημοσιεύτηκε το 1977 και είναι ο κρυπταλγόριθμος στον οποίο έχει γίνει η περισσότερη έρευνα σχετικά με την κρυπτογραφική του δύναμη. Ο DES είναι βασισμένος στον κρυπταλγόριθμο Lucifer της IBM, του οποίου το τμήμα του απλού κειμένου, του κρυπτοκειμένου, καθώς και το μέγεθος του κλειδιού είναι 128 bit.

  Ο κρυπταλγόριθμος DES έχει το χαρακτηριστικό ότι η κρυπτογράφηση και η αποκρυπτογράφηση μπορούν να υλοποιηθουν με την ίδια διαδικασία, με την μόνη διαφορά ότι το πρόγραμμα κλειδιού της αποκρυπτογράφησης παράγει

την αντίστροφη ακολουθία που παράγει το πρόγραμμα κλειδιού της κρυπτογράφησης [19]. Το μειονέκτημα του DES είναι ότι το πρόγραμμα κλειδιού του είναι αδύναμο, όχι μόνον επειδή από οποιοδήποτε κλειδί γύρου μπορεί να βρεθεί με σχετική ευκολία το αρχικό κλειδί, αλλά και επειδή υπάρχουν κλειδιά τα οποία παράγουν το ίδιο πρόγραμμα κλειδιού κατά την κρυπτογράφηση και την αποκρυπτογράφηση. Σε περίπτωση μάλιστα που το κλειδί είναι κάποιο από από τα κλειδιά του παρακάτω πίνακα, τότε το πρόγραμμα κλειδιών που προκύπτει είναι το ίδιο τόσο στην κρυπτογράφηση όσο και στην αποκρυπτογράφηση.

Αυτο σημαίνει ότι στην περίπτωση που ένα σύστημα χρησιμοποιεί διπλή κρυπτογράφηση, το κρυπτοκείμενο που προκύπτει θα είναι ίδιο με το απλό κείμενο. Τα κλειδιά αυτά ονομάζονται αδύναμα κλειδιά. Εκτός από τα αδύναμα κλειδιά υπάρχουν και τα ημιαδύναμα κλειδιά , τα οποία εμφανίζονται σε ζευγάρια όπου το πρόγραμμα κλειδιού του ενός είναι ισοδύναμο με το πρόγραμμα κλειδιού του άλλου με αντίστροφη σειρά. Έτσι η κρυπτογράφηση του ενός κλειδιού ακολουθούμενη από την κρυπτογράφηση του δεύτερου κλειδιού που ορίζουν ζευγάρι έχει ως αποτέλεσμα το κρυπτοκείμενο να είναι ίσο με το αρχικό απλό κείμενο.

- ο κρυπταλγόριθμος AES (Advanced encryption Standard).

  Ο AES προέκυψε από μια προσπάθεια αντικατάστασης του DES. Η προσπάθεια δημοσιοποιήθηκε το Σεπτέμβριο του 1997 με την προκήρυξη του Εθνικού Ινστιτούτου Προτύπων και Τεχνολογίας (National Institute of Standards and Technology, NIST). Ο AES είναι επαναληπτικός κρυπταλγόριθμος και βασίζεται σε κρυπτογράφηση γινομένου. Σε αντίθεση με τον DES η δομη του AES δεν είναι δίκτυο Feistel. Ένα δίκτυο Feistel κρυπτογραφεί ένα μέρος της εισόδου του σε κάθε γύρο (στην περίπτωση του DES τα μισά bits της εισόδου.). Στην περίπτωση του AES, σε κάθε γύρο κρυπτογραφείται όλη η είσοδος. Αυτό έχει ως αποτέλεσμα λιγότεροι αριθμοί γύρων. Ο AES μπορεί να αντισταθεί αποτελεσματικά σε όλες τις γνωστές κρυπταναλύτικες μεθόδους.

Κρυπταλγόριθμοι ροής

Οι τακτικές σχεδιασμού κρυπταλγόριθμων ροής μοιάζουν με αυτές των κρυπταλγόριθμων τμήματος. Αρχικά τίθενται τα κρυπτογραφικά κριτήρια που πρέπει να πληρεί ο υποψήφιος κρυπταλγόριθμος ροής και στη συνέχεια ακολουθεί ο σχεδιασμός και η ανάλυσή του. Τα κριτήρια που θα πρέπει να πληρεί ένας κρυπταλγόριθμος ροής είναι τα εξής:

- Η περίοδος της κλειδοροής θα πρέπει να είναι όσο το δυνατόν μεγαλύτερη.
- Τα *bit* της ακολουθίας της κλειδοροής θα πρέπει να περνούν με επιτυχία όλους τους γνωστούς ελέγχους περί τυχαιότητας.

- Το κλειδί του κρυπταλγόριθμου ροής καθορίζει την αρχική κατάσταση της γεννήτριας της κλειδοροής, που αντιστοιχεί στον προσδιορισμό της αρχικής θέσης στην περιοδική ακολουθία της κλειδοροής.

Από τους πιο διαδεδομένους κρυπταλγόριθμους ροής είναι ο RC4 ο οποίος χρησιμοποιείται σε πρωτόκολλα ασφαλείας (SSL) καθώς για μεγάλου μήκος κλειδιού θεωρείται ασφαλής. Οι σχετικά μικρές απαιτήσεις σε μνήμη και υπολογιστική ισχύ, καθιστούν τον $RC4$ μια από τις πρώτες επιλογές ως κρυπταλγόριθμο ροής.

Τυποποιημένοι Τρόποι Λειτουργίας

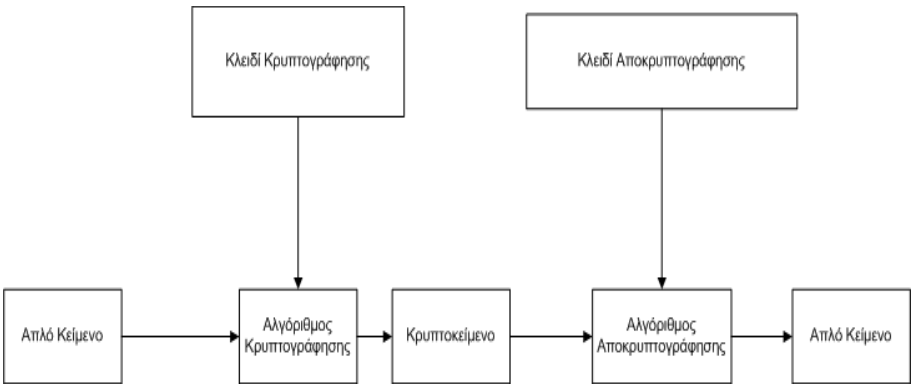
Οι κρυπταλγόριθμοι τμήματος συγκεκριμένης κρυπτογραφικής ισχύος διασυνδέονται κατάλληλα με στόχο την περαιτέρω αύξηση της δύναμης και την αποτελεσματικότερη απόκρυψη πιθανών υπολειμμάτων πληροφορίας του απλού κειμένου από το κρυπτοκείμενο. Στο πρότυπο FIPS 81 προτείνονται συγκεκριμένοι τρόποι διασύνδεσης (modes of operation) οι οποίοι προσδίδουν εγγυημένη κρυπτογραφική ισχύ. Οι 4 τυποποιημένοι τρόποι λειτουργίας κρυπταλγορίθμων τμήματος είναι:

1. Το Ηλεκτρονικό κωδικοβιβλίο (electronic codebook,ECB) όπου κάθε τμήμα p_i μεγέθους $n-bit$ του απλού κειμένου $P=[p_1p_2...p_l]$ τροφοδοτείται στον κρυπταλγόριθμο και το τμήμα κρυπτοκείμενου $c_i=e_k(p_i)$ μεγέθους $n-bit$ που προκύπτει στην έξοδο μεταδίδεται στον παραλήπτη.

2. Κρυπταλγόριθμος αλυσιδωτού τμήματος (cipher block chaining,CBC) όπου η κρυπτογράφηση ενός τμήματος του απλού κειμένου p_i εξαρτάται από το προηγούμενο τμήμα p_{i-1} μέσω της πράξεως $c_i = e_k(c_{i-1} \oplus p_i)$ ενώ η αποκρυπτογράφηση ορίζεται από την πράξη $p_i = d_k \oplus c_{i-1}$.

3. Ανάδραση κρυπταλγόριθμου (chipher feeback ,CFB) όπου το κρυπτοκείμενο προκύπτει από την επανακρυπτογράφηση του προηγούμενου κρυπτοκειμένου, συνδυασμένο με αποκλειστική διάζευξη με το απλό κείμενο. Έτσι η κρυπτογράφηση ορίζεται από την $c_i = e_k(c_{i-1}) \oplus p_i$ και η αποκρυπτογράφηση από την $p_i = e_k(c_{i-1}) \oplus c_i$. Η λειτουργία CFB μπορεί να θεωρηθεί ως κρυπταλγόριθμος ροής, όπου τα σύμβολα του απλού κειμένου είναι οι δυαδικές λέξεις $n\ bit$, που σημαίνει ότι το αλφάβητο του απλού κειμένου και του κρυπτοκειμένου αποτελείται από 2^n σύμβολα. Ο κρυπταλγόριθμος τμήματος λειτουργεί ως γεννήτρια κλειδοροής.

4. Ανάδραση εξόδου (output feeback, OFB) όπου η κρυπτογράφηση πραγματοποιείται με την αποκλειστική διάζευξη του απλού κειμένου με την ακολουθία της κλειδοροής. Αντίστοιχα η αποκρυπτογράφηση πετυχαίνεται με την αποκλειστική διάζευξη του κρυπτοκειμένου με την ακολουθία της κλειδοροής. Η διαδικασία δημιουργίας της κλειδοροής είναι ανεξάρτητη από το απλό κείμενο και το κρυπτοκείμενο.



Σχήμα 1.2: Αρχή λειτουργίας ασύμμετρου κρυπτοσυστήματος

1.1.2 Ασύμμετρα Κρυπτοσυστήματα

Η ασύμμετρη κρυπτογραφία βασίζεται αποκλειστικά σε μαθηματικές υποθέσεις για το λόγο ότι υπάρχει μαθηματική εξάρτηση μεταξύ του δημοσιού και του ιδιωτικού κλειδιού. Η ασφάλεια επομένως εξαρτάται από την πληροφορία που διαρρέει από το δημόσιο κλειδί, όσον αφορά το ιδιωτικό κλειδί.

Σε αντίθεση με τη συμμετρική κρυπτογραφία σε ένα ασύμμετρο κρυπτοσύστημα ο αλγόριθμος κρυπτογράφησης είναι ίδιος με με τον αλγόριθμο αποκρυπτογράφησης, με τη μόνη διαφόρα οτι κατά την αποκρυπτογράφηση χρησιμοποιείται το αντίστοιχο κλειδί όπως φαινεται στο Σχήμα 1.2. Έτσι σε ένα ασύμμετρο κρυπτοσύστημα όπου το ζευγάρι του δημοσίου και του ιδιωτικού κλειδιού είναι το (K_e, K_d), αν η κρυπτογράφηση ορίζεται απο την πράξη $c = e_{K_e}(m)$ τότε η αποκρυπτογράφηση ορίζεται από την $m = d_{K_d}(c) = d_{K_d}(e_{K_e}(x))$.

Λόγω συμμετρίας των πράξεων, αν η κρυπτογράφηση είναι η (e_{k_d}), τότε η αποκρυπτογράφηση θα ορίζεται από την (e_{k_e}). Η γνώση του δημοσίου κλειδιού από τον παραλήπτη είναι καθοριστικός παράγοντας για την εμπιστευτικότητα των μηνυμάτων που αποστέλλονται σε αυτόν. Η επίθεση του ενδιάμεσου ατόμου δείχνει την αδυναμία της ασύμμετρης κρυπτογραφίας να προσφέρει εμπιστευτικότητα στα επικοινωνούντα μέλη. Έτσι προκειμένου να προστατευτεί η εμπιστευτικότητα, αυτή παρέχεται με μηχανισμούς πιστοποίησης των δημοσίων κλειδιών. Οι μηχανισμοί επίδρασης του ασύμμετρου κρυπταλγόριθμου στο απλό κείμενο είναι ανάλογοι με αυτούς των συμμετρικών κρυπταλγόριθμων τμήματος. Έτσι και στην περίπτωση της ασύμμετρης κρυπτογραφίας, το απλό κείμενο χωρίζεται σε τμήματα συγκεκριμένου μήκους και το κάθε τμήμα κρυπτογραφείται ξεχωριστά. Αυτό δίνει τη δυνατότητα στον αντίπαλο να αναδιατάξει τα τμήματα του κρυπτοκειμένου, ώστε η αποκρυπτογράφηση να δώσει διαφορετικό μήνυμα από το αρχικό.

Η κρυπτογραφία παρέχει τα εργαλεία ώστε να προστατευτούν δύο ή περισσότερα επικοινωνούντα μέλη από αντιπάλους. Οι τρόποι και οι ευκαιρίες επίθεσης

του αντιπάλου καθορίζουν τις απαιτήσεις των μελών για προστασία. Οι ψηφιακές υπογραφές απευθύνονται σε κρυπτογραφικές υπηρεσίες, στην αυθεντικοποίηση και στη μη απάρνηση. Η αυθεντικοποίηση μπορεί να περιλαμβάνει ψηφιακές υπογραφές όπου ένα μέλος αποδεικνύει την ταυτότητα του έμμεσα δείχνοντας ότι γνωρίζει κάποια μυστική ποσότητα πληροφορίας, η οποία είναι συνδεδεμένη με την ταυτότητα του μέλους. Όπως αναφέραμε η αυθεντικοποίηση μπορεί να αναφερθεί στην αυθεντικοποίηση της ταυτότητας ενός μέλους, ή στην αυθεντικοποίηση ενός μηνύματος και παρέχει τη δυνατότητα ανίχνευσης των αυθαίρετων τροποποιήσεων ή επαναμεταδόσεων. Η ψηφιακή υπογραφή είναι επιθυμητή όταν δεν υπάρχει πλήρης εμπιστοσύνη μεταξύ του αποστολέα και του παραλήπτη, οπότε απαιτείται κάτι περισσότερο από αυθεντικοποίηση.

Ορισμός 1.1.1 *Έστω ένα σύνολο μηνυμάτων M, ένα σύνολο τιμών S και μια συνάρτηση μετασχηματισμού $S_A : M \longrightarrow S$, μιας οντότητας με ταυτότητα A. Το σύνολο S αποτελεί σύνολο υπογραφών, όταν μόνο η οντότητα A για οποιοδήποτε $m \in M$ μπορεί να υπολογίσει με σχετική ευκολία την: $S_A(m) = s \in S$. Η S_A ονομάζεται πράξη υπογραφής.*

Ορισμός 1.1.2 *Έστω ένα σύνολο μηνυμάτων M και έστω η δυαδική συνάρτηση $V_A : S \times M \longrightarrow \{0, 1\}$ όπου το 0 αντιστοιχεί στο ψευδές και το 1 αντιστοιχεί στο αληθές. Η συνάρτηση V_A ορίζει την πράξη επαλήθευσης αν μπορεί να υπολογιστεί σχετικά εύκολα από οποιονδήποτε έτσι ώστε για δεδομένο $(s, m) \in S \times M$ $V_A(s, m) = 1$ αν $S_A(m) = s$ και $V_A(s, m) = 0$ αν $S_A(m) \neq s$.*

Ορισμός 1.1.3 *Η πράξη ψηφιακής υπογραφής S_A, μαζί με την πράξη επαλήθευσης V_A, αποτελούν ένα σύστημα ψηφιακής υπογραφής για την οντότητα A.*

Το Κρυπτοσύστημα R.S.A

Ο R.S.A είναι ένας κρυπταλγόριθμος ασύμμετρου κλειδιού, το όνομα του οποίου προέρχεται από τους δημιουργούς του: Ron Rivest, Adi Shamir, Len Adleman. Ο R.S.A. βασίζεται στη δυσκολία παραγοντοποίησης μεγάλων αριθμών (σήμερα συνήθης τάξης $1024 - 2048$ bit). Χρησιμοποιούνται δύο κλειδιά, ένα δημόσιο κατά τη διάρκεια της κρυπτογράφησης και ένα ιδιωτικό για την αποκρυπτογράφηση.

Στην διαδικασία δημιουργίας των κλειδιών επιλέγονται αρχικά δύο μεγάλοι τυχαίοι πρώτοι αριθμοί p και q ίδιου μεγέθους και στη συνέχεια υπολογίζεται το

$$n = p \times q$$

και το

$$\phi(n) = (p - 1) \times (q - 1)$$

Επιλέγεται ένας ακέραιος $e > 1$ τέτοιος ώστε

$$(e,\ \phi(n)) = 1$$

και έπειτα υπολογίζεται ένας αριθμός d τέτοιος ώστε

$$d \times e \equiv 1 \; mod \; \phi(n)$$

Με το πέρας της διαδικασίας αυτής έχει σχηματισθεί τόσο το δημόσιο κλειδί που είναι το ζεύγος $(n,\ e)$ όσο και το ιδιωτικό κλειδί το οποίο είναι το ζεύγος $(n,\ d)$.

Το προς κρυπτογράφηση μήνυμα μπορεί να αντιπροσωπευθεί από έναν αριθμό m. Το κρυπτογραφημένο μήνυμα c υπολογίζεται με τον εξής τρόπο: $c = m^e \ (mod\ n)$

Κατά τη διαδικασία αποκρυπτογράφησης το κρυπτοκείμενο c υψώνεται στην d, δηλαδή:

$$\begin{aligned} c^d \ (mod\ n) &\equiv (m^e)^d \ (mod\ n) && (1.1)\\ &\equiv m^{e\times d} \ (mod\ n) && (1.2) \end{aligned}$$

Σύμφωνα με το μικρό θεώρημα του Fermat, για κάθε πρώτο αριθμό p και για κάθε αριθμό a με $0 < a < p$ ισχύει $a^{p-1} \equiv 1 \ (mod\ p)$. Επειδή όμως $e \times d \equiv 1 \ mod\ \phi(n)$ η σχέση 1.2 γίνεται

$$m^{e\times d} \equiv m^1 \equiv m \ (mod\ p)$$

Επειδή οι αριθμοί p και q είναι πρώτοι, σύμφωνα με το κινέζικο θεώρημα υπολοίπων έχουμε

$$m^{(q-1)\times(p-1)} \ (mod\ n) \equiv m \ (mod\ n)$$

Το κρυπτοσύστημα RSA μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός συστήματος ψηφιακών υπογραφών. Το σύστημα ψηφιακών υπογραφών RSA απαιτεί ότι όλες οι οντότητες έχουν στην κατοχή τους αντίστοιχα ζεύγη δημοσίου και ιδιωτικού κλειδιού.

Ορισμός 1.1.4 *Έστω p και q δύο πρώτοι αριθμοί και $n = p \times q$. Το σύστημα ψηφιακών υπογραφών RSA ορίζεται με $M = S = Z_n$ πράξη υπογραφής: $S_A(m) = m^{d_A} \ (mod\ n)$*
και πράξη επαλήθευσης: $V_A(s) = s^{e_A} \ (mod\ n)$ όπου $e_A \times d_A \equiv 1 \ (mod\ \phi(n))$, με e_A το δημόσιο κλειδί και d_A το ιδιωτικό κλειδί της οντότητας A.

Όσον αφορά το μέγεθος των RSA παραμέτρων, αν το κρυπτοσύστημα RSA χρησιμοποιείται μόνο για ψηφιακές υπογραφές και όχι για εμπιστευτικότητα, τότε ο δημόσιος εκθέτης e μπορεί να έχει οποιαδήποτε τιμή, καθώς δεν έχουν αναφερθεί αδυναμίες για μικρές τιμές του e. Ο αριθμός n θα πρέπει να έχει μέγεθος το λιγότερο ίσο με 1024 bits (δηλαδή το μέγεθος κάθε πρώτου αριθμού να είναι τουλάχιστον 512 $bits$), ενώ σε περιπτώσεις όπου απαιτείται μεγάλη διάρκεια ζωής των κλειδιών, προτείνεται το μέγεθος των 2048 $bits$ (οι πρώτοι αριθμοί εδώ θα πρέπει να είναι μεγέθους 1024 $bits$ ο καθένας).

Το Κρυπτοσύστημα EL-GAMAL

Το κρυπτοσύστημα El-Gamal παρουσιάστηκε για πρώτη φορά το 1985 από τον El-Gamal. Το κρυπτοσύστημα αυτό είναι δύο φορές πιο μεγάλο από το μήνυμα (βασικό μειονέκτημα έναντι του RSA. Η ασφάλεια του στηρίζεται στη δυσκολία επίλυσης προβλημάτων DPL (Discrete Logarithm Problem) δηλαδή για δοθέν $h = g^a mod\ p$, ο υπολογισμός του a είναι αδύνατος ακόμη κι αν ξέρουμε τα g, p, h όταν αυτά είναι πολύ μεγάλοι αριθμοί.

Κατά την διαδικασία της δημιουργίας των κλειδιών γίνεται επιλογή ενός μεγάλου τυχαίου πρώτου αριθμού p και ενός αριθμού a ο οποίος είναι πρωταρχική ρίζα του p. Στη συνέχεια γίνεται ο υπολογισμός του

$$a^d \equiv b(mod\ p)$$

όπου το d είναι μυστικό. Το δημόσιο κλειδί είναι το (a, b, p).

Για την κρυπτογράφηση γίνεται επιλογή ενός μυστικού κλειδιού. Στη συνέχεια υπολογίζονται τα:

$$t = m \times b^k (mod\ p)$$
$$r = a^k (mod\ p)$$

και έτσι προκύπτει το ιδιωτικό κλειδί $(r,\ t)$.

Η διαδικασία της αποκρυπτογράφησης έχει ως εξής: Γίνεται υπολογισμός του

$$t \times r^{-d} \equiv b^k \times m(a^k)^{-d} \equiv\ m\ (mod\ p)$$

Ψηφιακές υπογραφές στο κρυπτοσύστημα $El - Gamal$.

Η διαδικασία δημιουργίας της υπογραφής ενός μηνύματος έχει ως εξής: Αρχικά επιλέγεται ένας αριθμός j για τον οποίο ισχύει $(j,\ p-1) = 1$. Στη συνέχεια υπολογίζονται τα $r \equiv a^j (mod\ p)$, και $s \equiv j^{-1}(m - k \times r)(mod\ p-1)$.

Ο παραλήπτης του μηνύματος επιβεβαιώνει την ορθότητα της υπογραφής υπολογίζοντας αρχικά τα

$$V_1 \equiv b^r r^s (mod\ p)$$

και

$$V_2 \equiv a^m (mod\ p)$$

και στη συνέχεια εξετάζεται εάν ισχύει η

$$V_1 = V_2 (mod\ p)$$

Κρυπτοσύστημα Ελλειπτικών Καμπυλών

Τα κρυπτοσυστήματα που είναι βασισμένα στις ελλειπτικές καμπύλες προτάθηκαν το 1985 από τον Neal Koblitz και τον Victor Miller. Αυτά τα κρυπτοσυστήματα

(όπως και το κρυπτοσύστημα El-Gamal βασίζονται στο πρόβλημα του διακριτού λογαρίθμου. Αντί δακτυλιίων Z_n τα κρυπτοσυστήματα ελλειπτικών καμπυλών χρησιμοποιούν πεπερασμένα σώματα που έχουν τάξη κάποιον πρώτο αριθμό και βρίσκονται πάνω σε μια ελλειπτική καμπύλη.

Ορισμός 1.1.5 *Ελλειπτικές καμπύλες είναι το σύνολο των σημείων του επιπέδου, του οποίου οι συντεταγμένες ικανοποιούν την εξίσωση:* $y^2 = x^3 + a \times x + b$. *Οι παράμετροι a, b είναι ακέραιοι αριθμοί του τύπου* $4a^3 + 27b^2 \neq 0$

Η ελλεπτική καμπύλη αποτελείται από τις καμπύλες του σχήματος και επιπλέον από ένα σημείο Ο που ονομάζεται "σημείο στο άπειρο".

Ορισμός 1.1.6 *Έστω μια ελλειπτική καμπύλη ορισμένη στο* Z_r *και έστω* P *ένα σημείο της καμπύλης και ένα σημείο* Q *το οποίο αποτελεί βαθμωτό γινόμενο του* P. *Το πρόβλημα του διακριτού λογαρίθμου στην ελλειπτική καμπύλη είναι ο καθορισμός της λύσης για την οποία είναι* $n \times P = Q$.

Το σύνολο των απλών κειμένων καθώς και το σύνολο των κρυπτοκειμένων αποτελείται από τα σημεία μιας ελλειπτικής καμπύλης. Υπάρχουν αποτελεσματικοί αλγόριθμοι οι οποίοι αντιστοιχίζουν απλό κείμενο σε σημεία ελλειπτικης καμπύλης. Έτσι ένα απλό κείμενο εκφράζεται με τις συντεταγμένες ενός σημείου $P_m = (X_m,\ Y_m)$.

Έστω $G = (X_g,\ Y_g)$ ένα σημείο της ελλειπτικής καμπύλης από το οποίο θα προκύψει το ιδιωτικό και το δημόσιο κλειδί. Γίνεται επιλογή ακεραίου n_b το οποίο αποτελεί το ιδιωτικό κλειδί. Το δημόσιο κλειδί είναι το $(P_b,\ G,\ a,\ b)$ όπου $P_b = n_b \times G$. Στη συνέχεια επιλέγεται ένας ακέραιος K και υπολογίζεται η ποσότητα: $C_m = (K \times G,\ P_m + K \times P_b)$.

Από το ζεύγος σημείων που ορίζει το κρυπτοκείμενο, το πρώτο σημείο πολλαπλασιάζεται με το ιδιωτικό κλειδί της οντότητας και το αποτέλεσμα που προκύπτει αφαιρείται από το δεύτερο σημείο του ζεύγους. Έτσι μπορούμε να γράψουμε,

$$\begin{aligned} d_K(C_m) &= (P_m + K \times P_b) - (n_b \times K \times G) \\ &= (P_m + K \times P_b) - K \times P_b \\ &= P_m \end{aligned}$$

Οι ψηφιακές υπογραφές στις ελλειπτικές καμπύλες ή αλλιώς ECDSA (Elliptic Curve Digital Signature Algorithm) παρουσιάστηκαν πρώτη φορά από τον Scott Vanstone το 1992. Πρόκειται για ελλειπτικές καμπύλες ανάλογες του DSA των οποίων το ζεύγος των κλειδιών είναι συνδεδεμένο με ένα συνηθισμένο σύνολο παραμέτρων των ελλειπτικών καμπυλών $D = (q,\ FR,\ a,\ b,\ G,\ n,\ h)$. Για τον υπολογισμό της ψηφιακής υπογραφής αρχικά δημιουργούνται τα κλειδιά (δημόσιο και ιδιωτικό). Αρχικά επιλέγεται τυχαία ένας ακέραιος d στο διάστημα $[1,\ n-1]$ ο οποίος είναι το ιδιωτικό κλειδί και στη συνέχεια γίνεται ο υπολογισμός του

δημόσιου κλειδιού Q πολλαπλασιάζοντας το σημείο G της καμπύλης με το ιδιωτικό κλειδί, δηλαδή $Q = d \times G$.

Για να υπογραφεί το μήνυμα εκτελούνται τα εξής βήματα: Επιλέγεται τυχαία ένας ακέραιος k με $1 \leq k \leq n-1$ υπολογίζεται το $k \times G = (x_1,\ y_1)$ και γίνεται μετατροπή του x_1 σε ακέραιο αριθμό. Στη συνέχεια υπολογίζουμε το $r = x_1(mod\ n)$ και αν $r = 0$ επιστρέφουμε στο αρχικό βήμα διαφορετικά υπολογίζουμε το $k^{-1}(mod\ n)$, $SHA-1(m)$ και το $s = k^{-1}(e + d \times r)(mod\ n)$. Αν ισχύει όμως $s = 0$ επιστρέφουμε στο αρχικό βήμα διαφορετικά η υπογραφή είναι η $(r,\ s)$.

1.2 Αντικείμενο της Πτυχιακής

Στις προηγούμενες υποενότητες δόθηκε μια συνοπτική περιγραφή των βασικών αρχών των συμμετρικών και των ασύμμετρων κρυπτοσυστημάτων. Η παρεχόμενη από τα ασύμμετρα κρυπτοσυστήματα ασφάλεια καθορίζεται σε σημαντικό βαθμό από την επιλογή μεγάλων πρώτων αριθμών κατά τρόπο τυχαίο. Οι πρώτοι αριθμοί επίσης εμπλέκονται και σε προβλήματα όπως αυτό της εύρεσης του διακριτού λογάριθμου και της παραγοντοποίησης ακεραίων αριθμών τα οποία αποτελούν τους θεμέλιους λίθους των σύγχρονων ασύμμετρων κρυπτοσυστημάτων.

Δεδομένης της σημαντικότητας των πρώτων αριθμών αλλά και του γεγονότος ότι στη σημερινή εποχή η ασφαλής πρόσβαση σε δημόσια και ιδωτικά δίκτυα μέ τη χρήση μικρών σε μέγεθος, πόρους και επεξεργαστική ισχύ συσκευών είναι σχεδόν απαραίτητη, καθιστά αναγκαία την προσπάθεια να εξετασθούν οι πιο γνωστοί αλγόριθμοι πιστοποίησης πρώτων αριθμών σε τέτοιου είδους μικρές συσκευές οι οποίες ονομάζονται ενσωματωμένες συσκευές ή συστήματα.

Στην πτυχιακή αυτή εξετάζονται οι πιο γνωστοί αποδοτικοί αλγόριθμοι πιστοποίησης πρώτων αριθμών σε ενσωματωμένες συσκευές. Στο δέυτερο κεφάλαιο της πτυχιακής γινεται μια σύντομη εισαγωγή στη θεωρία πολυπλοκότητας προκειμένου να είναι κατανοητή η παρουσίαση της πολυπλοκότητας των αλγορίθμων πιστοποίησης πρώτων αριθμών που παρουσιάζονται στο τρίτο κεφάλαιο. Εκεί αριχκά εξετάζεται ο αιτιοκρατικός αλγόριθμος του Miller με πολυωνυμικό χρόνο εκτέλεσης ο πιθανοτικός αλγόριθμος των Solovay-Strassen και ο πιθανοτικός αλγόριθμος των Miller, Rabin και Shelfridge. Στο τέταρτο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα της ανάπτυξης των αλγορίθμων πιστοποίησης πρώτων αριθμών που εξετάσθηκαν στο κεφάλαιο 3 τόσο σε φορητό υπολογιστή όσο και ένα ασύρματο VoIP(Voice over IP τηλέφωνο. Αρχικά γίνεται μια σύντομη αλλά περιεκτική περιγραφή ενός ασύρματου VoIP τηλέφωνου βασισμένου στον επεξεργαστή AT76C902 της ATMEL. Στην συνέχεια παρασουσιάζεται η υλοποίηση σε γλώσσα C του αιτιοκρατικού αλγόριθμου του Miller, του πιθανοθεωρητικού αλγόριθμου των Solovay-Strassen και του αλγόριθμου των Miller, Rabin και Shelfridge καθώς και τα αποτελέσματα της εκτέλεσης των αλγορίθμων αυτών σε έναν φορητό υπολογιστή και στο ενσωματωμένο σύστημα της ATMEL AT76C902. Για την υλοποίηση αυτή χρησιμοποιήθηκε η βιβλιοθήκη αριθμητικής πολλαπλής ακρίβειας

GNU-MP αναλυτική περιγραφή της οποίας δίνεται στο παράρτημα.

Κεφάλαιο 2

Κλάσεις Υπολογιστικής Πολυπλοκότητας

2.1 Εισαγωγή στη Θεωρία Πολυπλοκότητας

Δεδομένου ενός επιλύσιμου υπολογιστικού προβλήματος το να συντάξουμε έναν αλγόριθμο που να το επιλύει είναι σε πολλές περιπτώσεις τετριμμένο. Άρκει απλώς να κατανοήσουμε το πρόβλημα και ένας αφελής (εξαντλητικός) αλγόριθμος καθίσταται προφανής. Ωστόσο ο αλγόριθμος αυτός είναι κατά κανόνα υπερβολικά αργός, και άρα ελάχιστα χρήσιμος.

Η θεωρία της υπολογιστικής πολυπλοκότητας χρησιμεύει για την διευκρίνηση των υπολογιστικών πόρων (πχ. χρόνος, χώρος, τυχαιότητα) που είναι απαραίτητοι για την επίλυση μιας συνάρτησης ή ενός προβλήματος. Κύρια αφορμή για την ανάπτυξη της είναι η ύπαρξη προβλημάτων που ενώ είναι επιλύσιμα δεν είναι γνωστός κάποιος αποδοτικός αλγόριθμος για την επίλυση τους. Αντικείμενο της θεωρίας πολυπλοκότητας είναι η μελέτη των υπολογιστικών προβλημάτων και η κατάταξη τους σε κλάσεις πολυπλοκότητας ανάλογα τους υπολογιστικούς πόρους που αυτά απαιτούν για να επιλυθούν αλγοριθμικά.

Ένα υπολογιστικό πρόβλημα είναι μια ερώτηση η οποία περιγράφεται από ένα σύνολο παραμέτρων ή αλλιώς ελεύθερων μεταβλητών καθώς και από τη σχέση που πρέπει να έχει η λύση του με τις μεταβλητές εισόδου. Είναι σημαντική η διάκριση μεταξύ του προβλήματος και ενός στιγμιότυπου του. Το στιγμιότυπο προκύπτει από μια απόδοση συγκεκριμένων τιμών στις παραμέτρους του προβλήματος. Η επίλυση ενός υπολογιστικού προβλήματος απαιτεί τη σχεδίαση ενός αλγορίθμου ή αλλιώς μιας σαφούς βήμα προς βήμα διαδικάσιας που να οδηγεί στη λύση σε πεπερασμένο αριθμό βημάτων.

Ο τρόπος εκτίμησης της πολυπλοκότητας ενός αλγορίθμου είναι γενικότερος.

Τις περισσότερες φορές η ακριβής εκτίμηση του αριθμού των βημάτων μπορεί να είναι πολύ δύσκολη ή και να εξαρτάται από τη συγκεκριμένη είσοδου του αλγορίθμου (δηλαδή από το συγκεκριμένο στιγμιότυπο του προβλήματος). Επίσης μπορεί και να μην αντανακλά απολύτως στο χρόνο που θα κάνει ένας υπολογιστής για να τρέξει, καθώς οι πράξεις σε μια πραγματική μηχανή δεν διαρκούν όλες τον ίδιο χρόνο. Η παράμετρος που προσδιορίζει το μέγεθος του προβλήματος εξαρτάται από τη φύση του προβλήματος που εξετάζουμε. Τα περισσότερα προβλήματα χαρακτηρίζονται από κάποια φυσική παράμετρο που μπόρει να αποτελέσει μέγεθος του προβλήματος.

2.1.1 Ασυμπτωτική Συμπεριφορά Αριθμητικών Συναρτήσεων

Στη θεωρία πολυπλοκότητας συχνά ενδιαφερόμαστε για την ασυμπτωτική συμπεριφορά της πολυπλοκότητας ενός υπολογιστικού προβλήματος ως μια συνάρτηση του μεγέθους της εισόδου ή κάποιων άλλων παραμέτρων. Τα παρακάτω ασυμπτωτικά όρια χρησιμοποιούνται συχνά στη θεωρία πολυπλοκότητας.

1. Ασυμπτωτικό πάνω όριο:

   Εάν υπάρχει μια θετική σταθερά c και ένας θετικός ακέραιος n_0 έτσι ώστε να ισχύει: $0 \leq f(n) \leq c \times g(n)$ για κάθε $n \geq n_0$ γράφουμε:

   $$f(n) = O(g(n))$$

   Εάν η $g(n)$ είναι σταθερά (δεν εξαρτάται δηλαδή από το n) τότε γράφουμε: $f(n) = O(1)$ ή για συντομία $f = O(1)$.

   Εάν ισχύει: $0 \leq f(n) < c \times g(n)$ για κάθε $n \geq n_0$ γράφουμε:

   $$f(n) = o(g(n))$$

2. Ασυμπτωτικό κάτω όριο:

   Εάν υπάρχει μια θετική σταθερά c και ένας θετικός ακέραιος n έτσι ώστε να ισχύει: $0 \leq c(g(n)) \leq f(n)$ για κάθε $n \geq n_0$ γράφουμε:

   $$f(n) = \Omega(g(n))$$

3. Ασυμπτωτικό σφιχτό όριο:

   Εάν υπάρχουν θετικές σταθερές c_1 και c_2 και ένας θετικός ακέραιος n_0 έτσι ώστε να ισχύει: $c_1 g(n) \leq f(n) \leq c_2 g(n)$ για κάθε $n \geq n_0$ γράφουμε:

   $$f(n) = \Theta(g(n))$$

Ορισμός 2.1.1 *Μια συνάρτηση $f(n)$ είναι αμελητaία εάν για κάθε σταθερά $c \geq 0$ υπάρχει ένας θετικός ακέραιος n_0 έτσι ώστε: $f(n) < n^{-c}$ για κάθε $n \geq n_0$.*

Για παράδειγμα $f(n) = o(n^{-c})$ για κάθε ακέραια σταθερά c. Με λίγα λόγια αυτό σημαίνει ότι η $f(n)$ μειώνεται (πλησιάζει προς το 0) γρηγορότερα από τον αντίστροφο κάθε πολυώνυμου και αυτό γιατί η $f(n)$ μειώνεται αυθαίρετα για αρκετά μεγάλα n. Για παράδειγμα κάθε πολυώνυμο $p(n)$ της συνάρτησης $f(n) = p(n) \setminus 2^n$ είναι αμελητέο.

Ορισμός 2.1.2 *Μια συνάρτηση $f(n)$ λέγεται μη αμελητέα εάν υπάρχει μία θετική σταθερά c και ένας θετικός ακέραιος n_0 έτσι ώστε: $f(n) > n^{-c}$ για κάθε $n > n_0$.*

Για παράδειγμα $f(n) = \Omega(n^{-c}))$. Στην πράξη μας ενδιαφέρει η εύρεση αποτελεσματικών αλγορίθμων οι οποίοι να υπολογίζουν μια συνάρτηση ή να επιλύουν ένα πρόβλημα. Συγκεκριμένα η έννοια της αποτελεσματικότητας σχετίζεται με τον χρόνο εκτέλεσης ενός αλγορίθμου. Ο χρόνος εκτέλεσης ενός αλγορίθμου (για συγκεκριμένου μεγέθους είσοδο) μπορεί να οριστεί ως ο αριθμός των βασικών πράξεων ή των βημάτων που θα χρειαστεί για να εκτελεσθούν. Σε κάθε περίπτωση ο χρόνος εκτέλεσης ενός αλγορίθμου μπορεί να μετρηθεί για την καλύτερη ή την χειρότερη περίπτωση.

Ο χρόνος εκτέλεσης χειρότερης περίπτωσης (worst-case running time) ενός αλγορίθμου είναι ένα ανώτερο όριο του χρόνου εκτέλεσης για κάθε είσοδο εκφραζόμενος ως μία συνάρτηση του μεγέθους της εισόδου.

Ο χρόνος εκτέλεσης μέσης περίπτωσης (average-case running time) ενός αλγορίθμου είναι ο μέσος χρόνος εκτέλεσης για κάθε είσοδο ενός συγκεκριμένου μεγέθους και εκφράζεται ως μια συνάρτηση του μεγέθους της εισόδου.

Στη σύγχρονη κρυπτογραφία συνηθίζεται να λέγεται ένας υπολογισμός αποτελεσματικός εάν αυτός ολοκληρώνεται σε πολυωνυμικό χρόνο για την χειρότερη περίπτωση (worst-case running time).

Ορισμός 2.1.3 *Μια συνάρτηση $f(n)$ ονομάζεται πολυωνυμική εάν ισχύει: $f(n) = O(n^c)$ για $c \in \aleph$. Σε αντίθετη περίπτωση ονομάζεται υπερπολυωνυμική.*

Ορισμός 2.1.4 *Ένας αλγόριθμος ονομάζεται αλγόριθμος πολυωνυμικού χρόνου εάν ο χρόνος εκτέλεσης χειρότερης περίπτωσης της συνάρτησής του είναι πολυωνυμικός για το ανάλογο μέγεθος εισόδου. Οι αλγόριθμοι για τους οποίους ο χρόνος εκτέλεσης δεν φράσσεται από ένα πολυώνυμο ονομάζονται υπερπολυωνυμικοί.*

Τα σημαντικότερα παραδείγματα υπερπολυωνυμικών αλγορίθμων είναι οι αλγόριθμοι εκθετικού χρόνου (exponential time algorithms). Ο χρόνος εκτέλεσης χειρότερης περίπτωσης αυτών των αλγορίθμων είναι $O(c^n)$.

Ορισμός 2.1.5 *Ένας αλγόριθμος ονομάζεται υποεκθετικού χρόνου (subexponential time algorithm) εάν ο χρόνος εκτέλεσης χειρότερης περίπτωσης είναι συνάρτηση της μορφής $e^{o(n)}$ όπου n είναι το μέγεθος της εισόδου.*

2.2 Υπολογιστικά Μοντέλα

Στην υπολογιστική θεωρία συνήθως χρησιμοποιούνται τα παρακάτω υπολογιστικά μοντέλα:

- Τα κυκλώματα *Bool* (Boolean circuits)
- Η μηχανή *Turing* (Turing machine)
- Οι μηχανές τυχαίας προσπέλασης (Random access machines)

Όλα τα παραπάνω υπολογιστικά μοντέλα είναι ισοδύναμα (δηλαδή εάν μια συνάρτηση (πρόβλημα) είναι υπολογίσιμη (επιλύσιμο) σ'ενα μοντέλο τότε είναι υπολογίσιμη (επιλύσιμο) και στα υπόλοιπα). Αυτό σημαίνει ότι όλες οι υπολογιστικές πολυπλοκότητες είναι ισοδύναμες (σε πολυωνυμικούς μετασχηματισμούς). Το μαθηματικό μοντέλο το οποίο έχει υιοθετηθεί στη Θεωρία Πολυπλοκότητας είναι η μηχανή Turing η οποία είναι ένα εντυπωσιακά απλό μοντέλο που αποδεικνύεται όμως ότι παρέχει όλη τη δυνατή υπολογιστικά ισχύ που μπορεί να έχει ο μηχανιστικός υπολογισμός και μάλιστα με επιδόσεις αρκετά κοντά σε αυτές των ρεαλιστικών υπολογιστών.

2.2.1 Η Μηχανή TURING

Η μηχανή *Turing* προτάθηκε από τον Alan Turing σαν ένα μοντέλο υπολογισμού και είναι μια μηχανή που επιδρά σε σύμβολα. Τα σύμβολα που μπορεί να διαχειριστεί μια μηχανή *Turing* είναι στοιχεία ενός πεπερασμένου συνόλου συμβόλων που ονομάζεται αλφάβητο. Μια ακολουθία από ένα σύνολο συμβόλων αποτελεί μια λέξη ή συμβολοσειρά. Τέλος ένα σύνολο συμβολοσειρών αποτελεί μια γλώσσα.

Τα σύμβολα της μηχανής μπορούμε να θεωρήσουμε ότι είναι γραμμένα σε μια ταινία που έχει άπειρο μήκος και προς τις δύο κατευθύνσεις (μοιάζει δηλαδή με τη γεωμετρική ευθεία). Η ταινία χωρίζεται σε κύτταρα, σε κάθε ένα από τα οποία μπορεί να εγγραφεί ένα σύμβολο. Η μηχανή μπορεί να διαβάζει ή να γράφει ένα σύμβολο σε κάθε κίνησή της μέσω μιας κεφαλής ανάγνωσης εγγραφής. Μια κίνηση της μηχανής *Turing* είναι συνδιασμός δύο πραγμάτων: του συμβόλου που βρίσκεται τη χρονική στιγμή πριν από την κίνηση στο κύτταρο κάτω από την κεφαλή και της τρέχουσας κατάστασης στην οποία βρίσκεται η μηχανή μας. Μια κατάσταση μπορεί να ειδωθεί σαν ένα στοιχείο μνήμης και κάθε μηχανή *Turing* έχει ένα πεπερασμένο σύνολο από καταστάσεις. Μια κίνηση της μηχανής *Turing* συνίσταται στην αλλαγή του συμβόλου που βρίσκεται κάτω από την κεφαλή με κάποιο άλλο, στην εν συνεχεία κίνηση της μια θέση αριστερά ή δεξία και στην αλλαγή της κατάστασής της.

Τα παραπάνω πρέπει φυσικά να ειδωθούν σαν μια "μηχανοποιημένη" περιγραφή της μηχανής *Turing*, η οποία είναι ένα μαθηματικό αντικείμανο, χωρίς ταινίες, κεφαλές κτλ.

Οι μηχανές *Turing* πολυωνυμικού χρόνου δουλεύουν με αιτιοκρατικό τρόπο, το οποίο σημαίνει ότι εκτελούν επανηλλειμένα ένα ή περισσότερα αιτιοκρατικά βήματα. Υπάρχουν δύο εναλλακτικοί τύποι μηχανής *Turing*:

- Η μη αιτιοκρατική μηχανή *Turing* (Nondeterministic Turing machine): Πρόκειται για μια πολυωνυμικού χρόνου μηχανή *Turing* η οποία δουλεύει με μη αιτιοκρατικό τρόπο. Η μη αιτιοκρατική μηχανή *Turing* μπορεί να επιλύσει ένα υπολογιστικό πρόβλημα εάν η λύση αυτή υπάρχει. Η διαφορά από τη συμβατική (αιτιοκρατική) μηχανή είναι ότι τώρα δίνουμε κατά κάποιον τρόπο στη μηχανή την ελευθερία να βρίσκεται ταυτόχρονα σε περισσότερα από ένα, σημεία του υπολογισμού. Η μη αιτιοκρατική μηχανή *Turing* είναι ένα θεωρητικό μη ρεαλιστικό μοντέλο του οποίου η αξία βρίσκεται στην επιπλέον δυνατότητα ανάλυσης της πολυπλοκότητας προβλημάτων που μας δίνει.

- Η πιθανοτική μηχανή *Turing* (Probabilistic Turing machine). Είναι μια πολυωνυμικού χρόνου μηχανή *Turing* η οποία δουλέυει με πιθανοτικό τρόπο. Σε αντίθεση με την μη αιτιοκρατική μηχανή *Turing* μία πιθανοτική μηχανή *Turing* μπορεί να "χτιστεί". Όπως και στην αιτιοκρατική μηχανή *Turing* μπορεί να έχει ένα πλήθος ταινιών. Μια από αυτές τις ταινίες ονομάζεται τυχαία ταινία και περιέχει ομοιόμορφα κατανεμημένα τυχαία σύμβολα. Είναι αξιόλογο το ότι υπάρχουν πολλα προβλήματα (τα οποία είναι αποτελεσματικά όσον αφορά το χρόνο και το χώρο) για τα οποία μπορεί να κατασκευαστεί αυτός ο τύπος της μηχανής *Turing*

2.2.2 Λειτουργία Μηχανής *TURING*

Η μηχανή *Turing* είναι μια εφτάδα $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ (με τα Q, Σ, Γ να είναι όλα πεπερασμένα σύνολα). Με Q συμβολίζουμε το σύνολο των καταστάσεων. Όπου Σ είναι το αλφάβητο των εισαγομένων input το οποίο δεν περιέχει τον ειδικό "κενό" χαρακτήρα και Γ είναι το αλφάβητο της ταινίας (ή αλφάβητο εργασίας) όπου το κενό ανήκει στο Γ και $\Sigma \subseteq \Gamma$. Με δ συμβολίζουμε τη συνάρτηση μετάβασης όπου: $Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$. Όπου q_0 είναι η αρχική κατάσταση της ΜΤ. Με q_{accept} συμβολίζουμε την κατάσταση αποδοχής (q_{accept} ανήκει στο Q), με q_{reject} την κατάσταση απόρριψης (ισχύει:$q_{reject} \neq q_{accept}$). Προφανώς ισχύει:$q_0$, q_{accept}, q_{reject} ανήκουν στο Q.

Όλες οι παραλλαγές της μηχανής *Turing* που παρουσιάσαμε χρησιμέυουν για να οριστούν οι κλάσεις πολυπλοκότητας. Εν συντομία η αιτιοκρατική μηχανή *Turing* χρησιμέυει για να ορίσουμε την κλάση πολυπλοκότητας P, η μη αιτιοκρατική για να ορίσουμε την κλάση πολυπλοκότητας NP και $coNP$ και η πιθανοτική για την κλάση πολυπλοκότητας PP και των υποκλάσεων αυτής.

2.3 Κλάσεις Πολυπλοκότητας

Μια κλάση πολυπλοκότητας καθορίζεται συνήθως από τα παρακάτω πέντε στοιχεία:

1. Κατηγορία υπολογιστικών προβλημάτων τα οποία περιέχει (απόφασης, αναζήτησης, βελτιστοποίησης)
2. Υπολογιστικό μοντέλο (πχ μηχανή $Turing$)
3. Τύπο υπολογισμού (αιτιοκρατικό ($deterministic$), μη αιτιοκρατικό ($non-deterministic$), πιθανοτικό τύπο ($probabilistic$)).
4. Υπολογιστικό αγαθό που αποτελεί το μέτρο πολυπλοκότητας. Κυριότερα αγαθά είναι ο χώρος και ο χρόνος.
5. Όριο για το υπολογιστικό αγαθό. Θα μπορούσε να είναι οποιαδήποτε ακέραια συνάρτηση.

Ορισμός 2.3.1 *Μια συνάρτηση $f : \aleph \longrightarrow \aleph \backslash \{0\}$ ονομάζεται κατάλληλη συνάρτηση πολυπλοκότητας αν ισχύουν τα παρακάτω:*

- *Για κάθε n, $f(n+1) \geq f(n)$*
- *Υπάρχει μια μηχανή $Turing$ M_f με είσοδο και έξοδο έτσι ώστε:*

$$M_f(x) = \sqcap f(x)$$

Η M_f λειτουργεί σε χρόνο $O(f(n)+n)$ και σε χώρο $O(f(n))$.

2.3.1 Βασικές Κλάσεις Πολυπλοκότητας

Μια γλώσσα έχει χωρική πολυπλοκότητα $f(n)$ εάν αποφασίζεται απο μια NMT (Ντετερμινιστική (ή αιτιοκρατική) Μηχανή $Turing$) με πολλές ταινίες με φράγμα χώρου $f(n)$. Για οποιαδήποτε συνάρτηση $f(n)$ ορίζουμε κλάσεις πολυπλοκότητας:
$DTIME(f(n)) = \{L/L$ έχει χρονική πολυπλοκότητα$f(n)\}$.
$DSPACE(s(n)) = \{L/L$ έχει χωρική πολυπλοκότητα $s(n)\}$

- $P = \cup_{i>0} TIME(n^i)$
- $NP = \cup_{i=0} NTIME(n^i)$
- $PSPACE = \cup_{i=0} SPACE(n^i)$
- $EXP = \cup_{i=0} TIME(2^{n^i})$
- $L = SPACE(\log n)$
- $NL = NSPACE(\log n)$

2.3.2 Η Κλάση P

Ορισμός 2.3.2 *Η κλάση υπολογιστικής πολυπλοκότητας P ("polynomial time") αναφέρεται στην κλάση των προβλημάτων απόφασης $D \subseteq \{0,1\}$ τα οποία είναι επιλύσιμα σε πολυωνυμικό χρόνο από μια ντετερμινιστική μηχανή Turing (για παράδειγμα υπάρχει μια ντετερμινιστική πολυωνυμικού χρόνου μηχανή Turing M για την οποία ισχύει $M(x) = 1$ αν και μόνο αν $x \in D$). Επομένως η κλάση πολυπλοκότητας P συμπεριλαμβάνει όλα τα προβλήματα τα οποία είναι επιλύσιμα ντετερμινιστικά σε πολυωνυμικό χρόνο.*

2.3.3 Η Κλάση NP

Ορισμός 2.3.3 *Η κλάση υπολογιστικής πολυπλοκότητας NP (Nondeterministic polynomial time) αναφέρεται στην κλάση των προβλημάτων απόφασης $D \subseteq \{0,1\}$ τα οποία είναι επιλύσιμα σε πολυωνυμικό χρόνο από μια μη-ντετερμινιστική μηχανή Turing. (Παράδειγμα: υπάρχει μια μη ντετερμινιστική πολυωνυμικού χρόνου μηχανή Turing M για την οποία ισχύει:$M(x) = 1$ αν και μόνο αν $x \in D$.) Με λίγα λόγια ένα πρόβλημα απόφασης είναι NP αν μια θετική απάντηση μπορεί να επαληθευτεί (τουλάχιστον) σε πολυωνυμικό χρόνο δίνοντας κάποιες επιπλέον πληροφορίες (και ονομάζεται certificate ή witness).*

2.3.4 Η Κλάση $coNP$

Ορισμός 2.3.4 *Η κλάση υπολογιστικής πολυπλόκότητας $coNP$ αναφέρεται στην κλάση των προβλημάτων απόφασης $D \subseteq \{0,1\}$ για τα οποία μία αρνητική απάντηση μπορεί να επαληθευτεί σε πολυωνυμικό χρόνο δίνοντας κάποιες επιπλέον πληροφορίες (ονομάζεται certificate ή μάρτυρας witness).*

Ισχύει ότι: $coNP \neq NP$

2.3.5 Η Κλάση PP

Ορισμός 2.3.5 *Η κλάση υπολογιστικής πολυπλοκότητας PP αναφέρεται στην κλάση των προβλημάτων απόφασης $D \subseteq \{0,1\}$ τα οποία είναι επιλύσιμα σε πολυωνυμικό χρόνο από μια πιθανοτική μηχανή Turing. Οι πιθανοτικές μηχανές Turing θέτουν σε εφαρμογή πιθανοτικούς (ή τυχαιοποίημένους) αλγόριθμους όπως οι αλγόριθμοι Monte Carlo, Las Vegas, Atlantic city.*

Παρουσίαση Τυχαιοποιημένων (Probabilistic) Αλγορίθμων

Το βασικό χαρακτηριστικό των τυχαιοποιημένων αλγορίθμων είναι ότι για την ίδια περίπτωση προβλήματος δίνουν διαφορετικό αποτέλεσμα εάν εκτελεσθούν δεύτερη

φορά. Ακόμη μπορεί να υπάρχει σημαντική διαφορά και στο χρόνο εκτέλεσης του αλγόριθμου. Οι τυχαιοποιημένοι αλγόριθμοι διακρίνονται σε τέσσερις κατηγορίες:

1. Αλγόριθμοι Monte Carlo
2. Αριθμιτικοί προσεγγιστικοί ή Αλγόριθμοι Atlantic city
3. Αλγόριθμοι Las Vegas
4. Αλγόριθμοι Sherwood

Οι αλγόριθμοι Monte Carlo δίνουν ένα αποτέλεσμα το οποίο δεν είναι πάντα προσέγγιση της ακριβούς λύσης. Το αποτέλεσμα που δίνει ένας αλγόριθμος Monte Carlo μπορεί να μην είναι επιτυχές, αλλά η πιθανότητα επιτυχίας του αλγορίθμου αυξάνεται ανάλογα με το χρόνο εκτελεσής του. Οι αλγόριθμοι Las Vegas δίνουν πάντοτε την ίδια (σωστή) έξοδο για τα ίδια δεδομένα εισόδου, απλώς ο χρόνος εκτέλεσης εξαρτάται από την έξοδο της γεννήτριας των τυχαίων αριθμών αλλά δεν δίνουν πάντοτε την ίδια έξοδο για τα ίδια δεδομένα εισόδου. Προφανώς η ιδιότητα αυτή δεν είναι επιθυμητή γιατί άλλοτε ο αλγόριθμος θα δίνει τη σωστή λύση και άλλοτε κάποια λανθασμένη. Με λίγα λόγια ένας Monte Carlo αλγόριθμος εάν δεν τερματίζει εντός ενός συγκεκριμένου διαστήματος αποτυγχάνει. Ομοίως με τους αλγορίθμους Monte Carlo ένας Atrlantic city αλγόριθμος τρέχει σε πολυωνυμικό χρόνο και έχει μόνο πιθανοτικά ή αποδεδειγμένα σωστά αποτελέσματα. Αντίθετα όμως με τους Monte Carlo κάθε φορά που οι Atlantic city αλγόριθμοι κάνουν λάθος τότε και οι δύο απάντήσεις (δηλ. ναι ή όχι) έχουν την ίδια πιθανότητα να είναι σωστές.

Παρακάτω θα παρουσιάσουμε κάποιους Monte Carlo αλγόριθμους πιστοποίησης πρώτων αριθμών. Δηλαδή αλγόριθμους που με σιγουριά μας δίνουν αποτέλεσμα σύνθετους αριθμούς και με πολύ μεγάλη πιθανότητα (όχι όμως με σιγουριά) πρώτους.

2.3.6 Η Κλάση ZPP

Ορισμός 2.3.6 *Η κλάση υπολογιστικής πολυπλοκότητας ZPP ("zero sided error probabilistic polynomial time") είναι μια υποκλάση της PP. Περιλαμβάνει όλα τα προβλήματα απόφασης $D \subseteq \{0,1\}$ για τα οποία υπάρχει μια πιθανοτική μηχανή Turing M τέτοια ώστε $\forall x \in \{0,1\}$ τότε:*

$$P_r[M \text{ outputs } Yes | x \in D] = 1$$

και

$$P_r[M \text{ outputs } Yes | x \notin D] = 0$$

Ο χαρακτηρισμός της λανθασμένης πιθανότητας σημαίνει ότι η πιθανοτική (πολυωνυμικού χρόνου) μηχανή Turing δεν μπορεί να κάνει κάποιο λάθος. Φαίνεται λοιπόν ότι η κλάση ZPP ισούται με την P. Όμως δεν είναι αυτή η υπόθεση

αφού υπάρχουν προβλήματα τα οποία μπορούν να επιλυθούν με τις πιθανοτικές μηχανές Turing πιο αποτελεσματικά απ'οτι με τις αιτιοκρατικές μηχανές (δουλεύοντας ταυτόχρονα σε πολυωνυμικό χρόνο).

2.3.7 Η Κλάση BPP

Ορισμός 2.3.7 *Η κλάση υπολογιστικής πολυπλοκότητας BPP ("bounded error probabilistic polynomial time") είναι μια υποκλάση της PP. Περιλαμβάνει όλα τα προβλήματα απόφασης $D \subseteq \{0,1\}$ για τα οποία υπάρχει μια πιθανοτική μηχανή Turing M τέτοια ώστε $\forall x \in \{0,1\}$ τότε:*

$P_r[M\ outputs\ Yes | x \in D] \geq \epsilon$
και $P_r[M\ outputs\ Yes | x \notin D] \leq \delta$ *όπου* $\epsilon \in (1/2,\ 1)$ *και* $\delta \in (0,\ 1/2)$

Σχέσεις Κλάσεων Πολυπλοκότητας

- $TIME(f(n)) \subseteq NTIME(f(n))$
- $SPACE(f(n)) \subseteq NSPACE(f(n))$
- $TIME(f(n)) = C_o TIME(f(n))$
- $SPACE(f(n)) = C_o SPACE(f(n))$
- $NTIME(f(n)) \subseteq \cup_{c>1} TIME(c^{f(n)})$
- $NSPACE(f(n)) \subseteq \cup_{c>1} TIME(c^{\log n + f(n)})$
- $NSPACE(f(n)) \subseteq SPACE(f^2(n)$, για $f(n) \geq \log n)$
- $NSPACE(f(n)) = C_o NSPACE(f(n))$, για $f(n) \geq \log n)$

Ορισμός 2.3.8 *Μια γλώσσα έχει χρονική πολυπλοκότητα $f(n)$ εάν αποφασίζεται από μια NMT με πολλές ταινίες με φράγμα χρόνου $f(n)$. Ο χώρος είναι πιο ισχυρός από το χρόνο αφού μπορούμε να επαναχρησιμοποιήσουμε το χώρο ενώ το χρόνο όχι.*

Για οποιαδήποτε NMT M μονής ταινίας εργασίας και οποιαδήποτε είσοδο X, $Space_X \leq Time_M(X) + 1$

Επομένως βάσει όλων των παραπάνω ισχύει ότι:

1. $L \subseteq NL$
2. $NL \subseteq P$

3. $P \subseteq NP$
4. $P \subseteq C_o NP$
5. $NP \subseteq PSPACE$
6. $C_o NP \subseteq PSPACE$
7. $PSPACE \subseteq NEXP$
8. $PSPACE = NPSPACE$
9. $P = C_o P$
10. $PSPACE = C_O PSPACE$
11. $NPSPACE = C_o NPSPACE$
12. $NL = C_o NL$
13. $P \subset EXP$
14. $NL \subset PSPACE$

Κεφάλαιο 3

Αλγόριθμοι Πιστοποίησης Πρώτων Αριθμών

Σε αυτό το κεφάλαιο παρουσιάζουμε αρχικά ένα άμεσο αποτέλεσμα ότι $PRIMES \in coNP$ και ότι $PRIMES \in NP$ (Pratt 1977). Στη συνέχεια αναλύουμε τον αλγόριθμο των Solovay Strassen (1977), τον αλγόριθμο του Miller (1975), ο οποίος βελτιώθηκε από τον Rabin (1980) και τον Selfridge. Αυτοί οι αλγόριθμοι τοποθέτησαν το πρόβλημα στο $coRP$. Στηρίζονται σε κάποιες απλές αλλά σημαντικές ιδιότητες των πρώτων αριθμών.

Μια γενίκευση της ιδέας του Miller το 1983 από τους Adleman, Pomernace, Rumely κατάφερε να μας δώσει έναν αιτιοκρατικό αλγόριθμο πιστοποίησης πρώτων αριθμών ο οποίος τρέχει σε χρόνο $(\log n)^{O(\log\log\log n)}$ αλλά είναι μη πρακτικός.

Το 1986, οι Goldwasser, Kilian παρουσίασαν έναν αλγόριθμο ο οποίος κάνει χρήση ελλειπτικών καμπυλών και τρέχει σε αναμενόμενο πολυωνυμικό χρόνο για σχεδόν όλες τις εισόδους. Σε αυτήν την ιδέα βασίστηκε ο *Atkin* για να παρουσιάσει έναν παρόμοιο αλγόριθμο την ίδια σχεδόν χρονιά ενώ οι Adleman και Huan το (1992) μετέτρεψαν τον αλγόριθμο των Goldwasser και Kilian για να κατασκευάσουν έναν αλγόριθμο ο οποίος τρέχει σε αναμενόμενο πολυωνυμικό χρόνο για όλες τις εισόδους. Απέδειξαν έτσι ότι $PRIMES \in ZPP$. Τέλος, το 2002 οι Agraval, Kayal και Saxena παρουσίασαν έναν αλγόριθμο που απαιτεί πολυωνυμικό χρόνο η εκτέλεση του, δηλαδή $PRIMES \in P$. Παρόλα αυτά, μέχρι σήμερα δεν έχει αξιοποιηθεί πρακτικά.

3.1 Το πρόβλημα της Πιστοποίησης των πρώτων αριθμών

Το πρόβλημα της πιστοποίησης πρώτου αριθμού, το να αποφανθούμε δηλαδή αν ένας αριθμός είναι πρώτος ή σύνθετος απασχολεί τους μαθηματικούς τόσο σε θεωρητική όσο και σε πρακτική σκοπιά. Όπως αναφέρθηκε και στο κεφάλαιο 1 ένας μεγάλος αριθμός ασύμμετρων κρυπτοσυστημάτων χρειάζεται μεγάλους πρώτους αριθμούς (π.χ στο κρυπτοσύστημα RSA οι πρώτοι αριθμοί είναι της τάξης των $1024 - 2048$ bit, στο κρυπτοσύστημα $El - Gamal$ της τάξης των $1024 - 2048$ bit και στα κρυπτοσυστήματα ελλειπτικών καμπυλών της τάξης των $160 - 224$ bit). Είναι φανερή λοιπόν η ανάγκη ένος αποτελεσματικού αλγόριθμου πιστοποίησης πρώτων αριθμών σε πολυωνυμικό χρόνο.

Συμβολίζουμε με PRIMES το σύνολο των πρώτων αριθμών και με COMPOSITES συμβολίζουμε το σύνολο των σύνθετων αριθμών.

Μια πρώτη προσέγγιση του προβλήματος προκύπτει άμεσα από τον ορισμό των πρώτων αριθμών. Για να ελέγξουμε αν ένας αριθμός n είναι πρώτος μπορούμε να διαιρέσουμε τον n με όλους τους αριθμούς που είναι μικρότεροι ή ίσοι του n. Αν κάποιος τον διαιρεί τότε $n \in COMPOSITES$ αλλιώς $n \in PRIMES$. Αυτός ο τρόπος ελέγχει όλους τους πρώτους μικρότερους ή ίσους από κάποιο n. Δεν είναι όμως αποδοτικός αφού για να ελέγξει κανείς εάν ένας αριθμός είναι πρώτος χρειάζεται ένας μεγάλος αριθμός βημάτων.

Μια άλλη γνωστή προσέγγιση στο πρόβλημα της πιστοποίησης των πρώτων αριθμών είναι το τετραγωνικό κόσκινο. Σκοπός της προσέγγισης αυτής είναι να βρούμε αριθμούς των οποίων το τετράγωνο είναι ίσο με με το γινόμενο κάποιων μικρών πρώτων. Το σύνολο αυτών των πρώτων αριθμών το ονομάζω βάση. Η ιδέα για την κατασκεύη του πίνακα είναι να δημιουργηθούν τέλεια τετράγωνα που να είναι λίγο μεγαλύτερα από ένα πολλαπλάσιο του n. Τότε τα τετράγωνα αυτά είναι "μικροί" αριθμοί $mod\ n$ δηλαδή γράφονται σαν γινόμενα μικρών πρώτων αριθμών.

Το τετραγωνικό κόσκινο βασίζεται στην εξής βασική πρόταση:

Πρόταση 3.1.1 *Έστω n ακέραιος και έστω ότι υπάρχουν ακέραιοι x, y έτσι ώστε να ισχύει:*

$$x^2 \equiv y^2 (mod\ n)$$

με

$$x \neq \pm y (mod\ n)$$

τότε ο n δεν είναι πρώτος και ο $d = (x - y, n)$ είναι διαιρέτης του n.

Έστω ότι θέλουμε να παραγοντοιήσουμε τον $n = 3837523$. Παρατηρώ ότι
$9398^2 \equiv 5^5 \times 19 (mod\ n) \Longrightarrow n_1$
$19095^2 \equiv 2^2 \times 5 \times 13 \times 19 (mod\ n) \Longrightarrow n_2$

$1964^2 \equiv 3^2 \times 13^3 (mod\ n) \Longrightarrow n_3$
$17078^2 \equiv 2^6 \times 3^2 \times 11 (mod\ n) \Longrightarrow n_4$.

Έχουμε ότι $n_1 \times n_2 \times n_3 \times n_4 \equiv (2^4 \times 3^2 \times 5^3 \times 11 \times 13 \times 19)^2$ όπου $n_1 \times n_2 \times n_3 \times n_4 = x^2$ και θέτουμε όπου $(2^4 \times 3^2 \times 5^3 \times 11 \times 13 \times 19)^2 = y$. Παρατηρούμε ότι $x \neq \pm y (mod\ n)$. Άρα από βασική πρόταση, ο n δεν είναι πρώτος άρα μπορούμε να παραγοντοποιήσουμε.

Ένας ακόμη απλός αλγόριθμος πιστοποίησης πρώτων αριθμών είναι η εξαντλητική αναζήτηση των διαιρετών ενός αριθμού n. Αρχίζοντας από το 2 εξετάζουμε κάθε ακέραιο $m < n$ αν είναι διαιρέτης του. Αν κάποιος είναι διαιρέτης του τότε σταματάμε και ο n δεν είναι πρώτος. Διαφορετικά αν εξαντληθούν όλοι οι ακέραιοι $m < n$ χωρίς να βρεθεί διαιρέτης τότε ο n είναι πρώτος.

Επιπλέον βασιζόμενοι στην παρατήρηση ότι κανένας αριθμός n δεν έχει διαιρέτη μεγαλύτερο του $n/2$ τροποποιούμε τον παραπάνω αλγόριθμο εξετάζοντας όλους τους αριθμούς m έτσι ώστε $m < (n/2)$.

Παρατηρούμε όμως, ότι αν ένας αριθμός n δεν είναι πρώτος τότε έχει τουλάχιστον δύο διαιρέτες μεγαλύτερους του 1. Σάυτην την περίπτωση τουλάχιστον ένας διαιρέτης είναι μικρότερος από την τετραγωνική ρίζα του αριθμού. Τροποποιούμε τον προηγούμενο αλγόριθμο εξετάζοντας όλους τους αριθμούς m που είναι μικρότεροι από την τετραγωνική ρίζα του n, αν η τελευταία δεν είναι ακέραιος αριθμός. Αλλιώς ο αριθμός δεν είναι πρώτος, επειδή τον διαιρεί και η τετραγωνική του ρίζα.

Αλγοριθμος Γλωσσομάθεια

Αλγόριθμος Π
  Μεταβλητές
    $integer : N$
  $Read\ N$
  $While\ N \leq 100$
    $If\ ((N-1)! + 1)\ \mathrm{mod}\ N) == 0$
      $Write$ “ είναι πρώτος ”
    $else\ Write$ “ δεν είναι πρώτος ”

3.1.1 Το Μικρό Θεώρημα του Fermat

Μέχρι τον 17ο αιώνα δεν υπήρχε κάποιος αποτελεσματικός αλγόριθμος πιστοποίησης πρώτων αριθμών και αυτό γιατί όλες οι διαδικασίες που χρησιμοποιούσαν οι μαθηματικοί ήταν ασύμφορες χρονικά. Τον 17ο αιώνα όμως ένας Γάλλος μαθηματικός ο *Pierre de Fermat* απέδειξε το παρακάτω θεώρημα:

Θεώρημα 3.1.1 *Για κάθε πρώτο αριθμό n και για κάθε αριθμό a (με $0 < a < n$) ισχύει:*

$$a^{n-1} \equiv 1 (mod\ n)$$

Αλγόριθμος του $Fermat$

$for\ i = 1\ to\ k$
$choose\ a \in [2, n-1]$
$gcd(a, n) = 1$
$q \equiv a^{p-1}(mod\ n)$
$if\ (q \neq 1) \longrightarrow$ "$composite$"
$else$ "$probably\ prime$"

Η πολυπλοκότητα του αλγορίθμου του $Fermat$ είναι:

$$O(klog^2 n\ log\ (log\ n)\ log(log(log\ n)))$$

Πόσο σίγουροι όμως μπορούμε να είμαστε για την ακρίβεια των αποτελεσμάτων·

3.1.2 Ψευδοπρώτοι (Pseudoprimes)

Με μια γρήγορη ματιά παρατηρούμε ότι ο αλγόριθμος του Fermat επιστρέφει τα εξής αποτελέσματα: Composite στην περίπτωση που ο αριθμός είναι σύνθετος ή Probably Prime στην περίπτωση που ο αριθμός δεν είναι σύνθετος. Αυτό συμβαίνει γιατί υπάρχουν αριθμοί οι οποίοι στην πραγματικότητα είναι σύνθετοι άλλα αν τους "τρέξουμε" στον αλγόριθμο του Fermat θα μας δώσει ως αποτέλεσμα ότι είναι πρώτοι. Αυτοί οι αριθμοί ονομάζονται ψευδοπρώτοι. Με λίγα λόγια ψευδοπρώτοι είναι αριθμοί αυτοί οι οποίοι μπορούν να κοροιδέψουν το μικρό θεώρημα του Fermat.

Ορισμός 3.1.1 *Ένας σύνθετος περιττός αριθμός n για τον οποίο ισχύει:*

$$a^{n-1} \equiv 1(mod\ n)$$

ονομάζεται ψευδοπρώτος για την βάση a.

Το σύνολο όλων των ψευδοπρώτων συμβολίζεται $psp(a)$. Το σύνολο $psp(2)$ έχει εξετασθεί από πολλούς συγγραφείς. Για παράδειγμα γνωρίζουμε οτί υπάρχουν μόνο 5597 τέτοιοι αριθμοί τάξης 10^9 με μικρότερο τον 341.

Ορισμός 3.1.2 *Έστω n ένας σύνθετος αριθμός τέτοιος ώστε*

$$a^{n-1} \equiv 1(mod\ n)$$

για κάθε $a \in Z_n*$ τότε το n ονομάζεται αριθμός του Carmichael.

Υπάρχουν μόνο 8241 τέτοιοι αριθμοί με μικρότερο τον 561.

Ορισμός 3.1.3 *Έστω p ένας περιττός πρώτος και a ακέραιος αριθμός. Το a ορίζεται ως δευτεροβάθμιο υπόλοιπο (quadratic residue) modulo p αν $a \neq 0(mod\ p)$ και η ισότητα $y^2 \equiv a(mod\ p)$ έχει λύση $y \in Z$.*

Το a ορίζεται ως μη δευτεροβάθμιο υπόλοιπο *modulo* p αν $a \neq 0(mod\ p)$ και το a δεν είναι δευτεροβάθμιο υπόλοιπο *modulo* p.

Ο Euler χρησιμοποίωντας το μικρό θεώρημα του Fermat απέδειξε το παρακάτω θεώρημα το οποίο είναι γνωστό και ως Κριτήριο του Euler.

Θεώρημα 3.1.2 *Εάν a είναι δευτεροβάθμιο υπόλοιπο (quadratic residue) modulo p και*

$$a^{(p-1)} \equiv y^{(p-1)/2} \equiv 1\ (mod\ p)$$

τότε ο p είναι πρώτος.

Απόδειξη:

Υποθέτουμε ότι $a \equiv y^2(mod\ p)$. Έστω ότι p είναι πρώτος. Άρα ισχύει ότι: $a^{(p-1)} \equiv 1(mod\ p)$ για κάθε $a \neq 0(mod\ p)$. Επομένως έχουμε:

$$\begin{aligned} a^{(p-1)/2} &\equiv (y^2)^{(p-1)/2}(mod\ p) \\ &\equiv y^{p-1}(mod\ p) \\ &\equiv 1(mod\ p) \end{aligned}$$

Τέλος υποθέτουμε πως $a^{(p-1)/2} \equiv 1(mod\ p)$ και έστω b πρωταρχική ρίζα *modulo* p έτσι ώστε: $a \equiv b^i(mod\ p)$ για κάποιον θετικό ακέραιο i. Έτσι έχουμε:

$$\begin{aligned} a^{(p-1)/2} &\equiv (b^i)^{(p-1)/2}(mod\ p) \\ &\equiv b^{i(p-1)/2}(mod\ p) \end{aligned}$$

Αφού το b έχει τάξη $p-1$ πρέπει το $p-1$ να διαιρεί το $i(p-1)/2$. Ως εκ τούτου το i είναι άρτιος και η τετραγωνική ρίζα του a είναι $\pm b^{i/2}(mod\ p)$.

Πόρισμα 3.1.1 *Ένας περιττός σύνθετος αριθμός n για τον οποίο ισχύει $gcd(a,\ n) = 1$ και $a^{n-1} \equiv (a/n)(mod\ n)$ ονομάζεται Euler ψευδοπρώτος (Euler Pseudoprime) για τη βάση a.*

Το σύνολο όλων αυτών των περιττών σύνθετων n συμβολίζεται με $epsp(a)$.

3.1.3 Πολυπλοκότητα Προβλήματος Πιστοποίησης Πρώτων Αριθμών

Στο εδάφιο αυτό θα δείξουμε ότι το πρόβλημα πιστοποίησης πρώτων αριθμών είναι στην κλάση $NP \cap coNP$.

Θεώρημα 3.1.3 $PRIMES \in coNP$

Απόδειξη:

Αρκεί να δείξουμε ότι το συμπληρωματικό σύνολο του PRIMES το οποίο είναι το $COMPOSITES \in NP$. Είναι έυκολο να δούμε ότι $COMPOSITES \in NP$. Αν η είσοδος είναι ένας αριθμός $n > 1$ τότε μαντεύουμε 2 ακεραίους $m,\ d$ έτσι ώστε $1 < m$ και $d < n$ και ελέγχουμε εάν ισχύει: $n = m \times d$. Αν ναι τότε αποδεχόμαστε το n.

Θεώρημα 3.1.4 $PRIMES \in NP$

Απόδειξη:

Θα δείξουμε αρχικά ότι ο n είναι πρώτος αν $n \neq 1$ και η Z περιέχει ένα στοίχειο τάξης $n-1$.

Έστω ότι ο n είναι πρώτος. Τότε $n \neq 1$ και η Z είναι κυκλική. Επομένως η Z περιέχει ένα στοιχείο τάξης $\varphi(n) = n-1$. Αντίστροφα αν $n \neq 1$ τότε η τάξη του Z είναι τουλάχιστον $n-1$. Τότε θα πρέπει να έχουμε $\varphi(n) = n-1$ οπότε ο n είναι πρώτος, (το να ελέγξουμε αν ένα $g \in Z$ είναι τάξης $n-1$ σημαίνει να ελέγξουμε ότι $g^{n-1} \equiv 1\ (mod\ n)$, το οποίο μπορεί να γίνει σε πολυωνυμικό χρόνο, αλλά επίσης και ότι $g^i \equiv 1\ (mod\ n),\ \forall i = 1,\ 2,\ 3,\ ...,\ n-2$).

Έστω τώρα ότι $n > 1$. Το στοιχείο g είναι τάξης $n-1$ στην Z αν και μόνο αν:

$$g^{n-1} \equiv 1\ (mod\ n)$$

και

$$g \equiv 1\ (mod\ n)$$

$\forall$ πρώτο p που διαιρεί το $n-1$. Η αναγκαιότητα της συνθήκης είναι ξεκάθαρη. Έστω $g \in Z$ έτσι ώστε: $g^{n-1} \equiv\ 1(mod\ n)$ για κάθε πρώτο p που διαιρεί το $n-1$. Αν m είναι η τάξη του g στην Z, τότε πρέπει το m να διαιρεί το $n-1$. Αν $m < n-1$ υπάρχει κάποιος πρώτος p ο οποίος διαιρεί το $n-1$. Έστω λοιπόν $1 = md$ τότε $g \equiv (g^m)^d \equiv 1\ (mod\ n)$ το οποίο είναι άτοπο λόγω της υπόθεσης μας για το g. Οπότε $m = n-1$ και το g έχει τάξη $n-1$ στην Z. Οπότε αν ξέρουμε όλους τους πρώτους παράγοντες του $n-1$ μπορούμε να ελέγξουμε ότι το g είναι τάξης $n-1$ σε πολυωνυμικό χρόνο, λόγω του ότι το πολύ διαφορετικοί πρώτοι διαιρούν το $n-1$.

Αυτό μπορεί να φαίνεται ότι δεν κάναμε μεγάλη πρόοδο, μειώσαμε και τον έλεγχο πρώτου αριθμού στην εύρεση παραγοντοποίησης σε πρώτους παράγοντες. Ωστόσο η δυναμική του να δουλεύουμε μη-αιτιοκρατικά έρχεται να λύσει το πρόβλημα. Θα κατασκευάσουμε μια μη-αιτιοκρατική διαδικασία για να πιστοποιήσουμε πρώτους.

Συνεπώς, με δεδομένο ένα n, επέλεξε τυχαία έναν "γεννήτορα" g και μια παραγοντοποίηση σε πρώτους αν είναι πράγματι πρώτος, και μετά έλεγξε αν το g είναι τάξης $n-1$ στη Z. Αν όλα γίνουν επιτυχώς αποδεχόμαστε το n σαν πρώτο αλλιώς το απορρίπτουμε.

Επομένως απο τα δύο παραπάνω θεωρήματα προκύπτει ότι: $PRIMES \in NP \cap coNP$.

3.1.4 Ο Αιτιοκρατικός Αλγόριθμος του Miller

Χρησιμοποιώντας την εκτεταμένη υπόθεση του Riemmann (Extended Riemmann's Hypothesis (ERH)) ο Gary L. Miller έφτιαξε έναν αιτιοκρατικό αλγόριθμο πιστοποίησης πρώτων αριθμών. Γνωρίζοντας ότι υπάρχει αποτελεσματικός αλγόριθμος πιστοποίησης πρώτων αριθμών ο οποίος εκτελείται σε $O(n^{134})$ βήματα και εάν προσθέσουμε και την εκτενή υπόθεση του Riemann προκύπτει το παρακάτω θεώρημα:

Θεώρημα 3.1.5 *Υπάρχει αλγόριθμος πιστοποίησης πρώτων αριθμών ο οποίος εκτελείται σε*

$$O(\mid n \mid^4 \ log\ log\ \mid n \mid)$$

βήματα.

Ορισμός 3.1.4 *Έστω $n = p_1^{v_1}...p_m^{vm}$ μία παραγοντοποίηση σε πρώτους αριθμούς του περιττού αριθμού n. Ονομάζουμε παραγοντοποίηση σε πρώτους τη συνάρτηση η οποία αντιστοιχίζει τους φυσικούς αριθμούς σε κατάλληλη κωδίκευση των πρώτων παραγόντων και των εκθετών τους. Επίσης υπολογίζουμε τις παρακάτω τρεις συναρτήσεις:*

- $\varphi(n) = (p_1)^{v_1-1}(p_1-1)...p_m^{v_m-1}(p_m-1)$ *Euler's Function*
- $\lambda(n) = lcm\ (p_1)^{v_1-1}(p_1-1),\ ...,\ p_m^{v_m-1}(p_m-1)$ *(The Carmichael λ-function),*
- $\lambda'(n) = lcm\ p_1-1,\ ...,\ p_m-1$

Θεώρημα 3.1.6 *Οι συναρτήσεις φ, λ, λ΄ και η παραγοντοποίηση σε πρώτους είναι όλες ισοδυναμίες πολυωνυμικού χρόνου. Το θεώρημα αυτό βασίζεται στην εκτεταμένη υπόθεση του Riemann (Extended Riemann's Hypothesis (ERH)).*

Θεώρημα 3.1.7 *Extended Riemmann's Hypothesis (ERH): Τα μηδενικά στοιχεία της συνάρτησης Dirichlet L(S,χ), στην κρίσιμη λωρίδα* $0 \leq$*(πραγματικό μέρος του S)* ≤ 1 *βρίσκονται όλα στην γραμμή (πραγματικό μέρος του S)* $= 1/2$*, όπου το* χ *είναι οποιοσδήποτε από τους τρεις χαρακτήρες:*

- *Legendre symbol:*(a/p)
- *Jacobi symbol:*$(a/(p \times q))$
- $\chi(a) = e(2\pi i(ind_p a)/q)$ *εάν* $(a,\ p) = 1$ *ή* $\chi(a) = 0$ *εάν* $(a,\ p) \neq 1$

και $L(S,\chi) = \sum \chi(n)/n^s$ *από* $n - 1$ *έως* ∞

Ο *Miller* χρησιμοποιώντας τα παραπάνω θεωρήματα (3.1.5−3.1.7) προσπάθησε να φτιάξει έναν αποτελεσματικό αλγόριθμο πιστοποίησης πρώτων. Αποτέλεσμα αυτής της προσπάθειας είναι ο παρακάτω αλγόριθμος.

Αλγόριθμος του *Miller*

Ορισμός 3.1.5 *Έστω f μια υπολογίσιμη συνάρτηση φυσικών αριθμών, ορίζουμε το σύνολο* A_f *με είσοδο n ως:*

1. Ελέγχουμε εάν το n είναι τέλεια δύναμη, (για παράδειγμα εάν $n = m^s$ με $s \geq 2$). Εάν το n είναι τέλεια δύναμη τότε ο αλγόριθμος επιστρέφει "n is COMPOSITE" και σταματάει.
2. Όσο το $a \leq f(n)$ εκτελούνται τα παρακάτω βήματα. Έαν ισχύει κάποιο από τα παρακάτω τότε ο αλγόριθμος επιστρέφει "n is COMPOSITE" και σταματάει.
   - $a \mid n$.
   - $a^{n-1} \neq 1 (mod\ n)$.
   - $((a^{(n-1)/2^k} mod\ n) - 1,\ n) \neq 1$ για κάποιο k με $1 \leq k \leq 2^{n-1}$
3. Αλλιώς ο αλγόριθμος επιστρέφει "n is PRIME" και σταματάει.

Πρόταση 3.1.2 *Το σύνολο* A_f *ορίζεται παραπάνω ως η απλούστερη εκδοχή του αλγορίθμου που χρησιμοποιεί το θεώρημα* 3.1.5*. Το* A_f *παράγει τον αλγόριθμο πιστοποίησης πρώτων αριθμών σε* $O(\mid n \mid^5 log^2 \mid n \mid)$ *βήματα.*

1. Καταρχήν ο A_f πρέπει να ελέγξει εάν το n είναι μία τέλεια δύναμη.
2. Ο A_f πρέπει να ελέγξει τα βήματα $1 - 3$ για $f(n)$ διαφορετικών a.

- Το βήμα 1 χρειάζεται $O(|n|^2)$ βήματα
- Το βήμα 2 χρειάζεται $O(|n|M(|n|))$ βήματα
- Το βήμα 3 χρειάζεται $O((|n|M(|n|) + |n|^2)|n|))$ βήματα μέχρις ώτου ο gcd να υπολογιστεί σε $O(|n|^2)$ βήματα και $1 < k < |n|$, ο πολλαπλασιασμός παίρνει τουλάχιστον $|n|$ βήματα και επομένως το βήμα 3 χρειάζεται συνολικά $O(|n|^2M(|n|))$ βήματα.

Επομένως ο A_f τρέχει σε $O(|n|^4M(|n|))$ βήματα όπου

$$M(|\ n\ |) = O(|\ n\ |\ \ log\ |\ n\ |\ log\ log\ \ |\ n\ |)$$

άρα η πολυπλοκότητα του αλγορίθμου είναι:

$$O(|\ n\ |^5\ log\ \ |\ n\ |\ \ log\ log\ |\ n\ |)$$

Κατάρχήν πρέπει να σημειωθεί ότι το σύνολο A_f δεν είναι απαραίτητα διάφορο από όλους τους αριθμούς οι οποίοι είναι $\leq f(n)$ αλλά μόνο από όλους τους πρώτους αριθμούς οι οποίοι είναι $\leq f(n)$.

Ο αριθμός των βημάτων μέχρις ότου κάποιος πρώτος να ειναι $\leq f(n)$ είναι ίσος με:

$$O(f(n)/log\ f(n))$$

Από το θεώρημα των πρώτων αριθμών προκύπτει ότι ο ανώτερος αριθμος βημάτων για το θεώρημα 3.1.5 είναι ισος με:

$$O(|\ n\ |^4\ log\ log\ |\ n\ |)$$

για το λόγο αυτό ο Miller προσπάθησε να βελτιστοποιήσει το σύνολο A_f. Αποτέλεσμα αυτής της προσπάθειας είναι ο παρακάτω αλγόριθμος:

Ορίζουμε το σύνολο A_f ως εξής: Υπολογίζουμε το $p_1,\ ...,\ p_m$ όπου p_i είναι ο $i-$οστος πρώτος αριθμός και m τέτοιο ώστε: $p_m \leq f(n) < p_{m+1}$. Υπολογίζουμε τα Q, S έτσι ώστε: $n-1 = Q2^S$ όπου Q περιττός. Έστω $i = 1$ τότε:

1. Αν $i < m$ τότε $i = i + 1$. Αν $i = m$ τότε επιστρέφει "n is PRIME" και ο αλγόριθμος σταματάει.
2. Εάν $a \mid n$ τότε επιστρέφει "n is COMPOSITE" και σταματάει. Αλλιώς υπολογίζει τα $a^Q mod\ n$, $a^{Q2} mod\ n$,...,$a^{Q2^S} mod\ n$.
3. Εάν $a^{Q2^S} mod\ n \neq 1$ τότε επιστρέφει "n is COMPOSITE" και ο αλγόριθμος σταματάει.
4. Εάν $a^Q mod\ n = 1$ επιστρέφει στο βήμα α. Θέτουμε στο $J = max(J : a^{Q2^J} mod\ n \neq 1$
5. Εάν $a^{Q2^J} mod\ n = n - 1$ επιστρέφει στο βήμα α.

6. Επιστρέφει "n is COMPOSITE" και ο αλγόριθμος σταματάει.

Επομένως ο προηγούμενος αλγόριθμος του μετά τις αλλαγές που πραγματοποίησε ο *Miller* έχει την εξής μορφή:

Miller(n, k)
If neίναι τέλεια δύναμη $\longrightarrow$ *composite*
$n-1 = 2^s m$ (m is odd)
$f = min\{\lceil klog^2 n \rceil, n\}$
$\forall$πρώτο αριθμό $a \leq f$
{
$if\ a \mid n$ *return composite*
else if $a^{n-1} \neq 1(mod\ n)$ return composite
else
$if\ a^m \neq 1(mod\ n)$ {
βρες το $max\ j < s : (a^m)^{2^j} \neq 1(mod\ n)$
$if\ (a^m)^{2^j} \neq -1(mod\ n)$ return composite
}
return πρώτος

Σημείωση: Ο αλγόριθμος τρέχει σε χρόνο

$$O(\mid n \mid^4 log\ log \mid n \mid)$$

Για να αποδείξουμε ότι το A_f είναι αλγόριθμος πιστοποίησης πρώτων αριθμών πρέπει απλώς να επανεξετάσουμε την παρακάτω υπόθεση:

Υπόθεση 3.1.1 *$\lambda'(n) \mid n-1$ και ο n δεν είναι πρώτη δύναμη.*

3.2 Πιθανοτικοί Αλγόριθμοι Πιστοποίησης Πρώτων Αριθμών

Το 1957 ο *Robinson* στην προσπάθεια του να βρει έναν αποτελεσματικό αλγόριθμο εύρεσης πρώτων αριθμών ο οποίος να εκτελείται σε πολυωνυμικό χρόνο χρησιμοποίησε το κριτήριο του *Euler*. Απέδειξε ότι αν ισχύει $(b,\ N) = 1$ τότε η υπόθεση

$$b^{(N-1)/2} \equiv (b/N)(mod\ N)$$

ισχύει (όπου $(\cdot/N)$ είναι το Jacobi symbol και το οποίο θα μπορούσε να χρησιμοποιήθει για να γίνει ο ελεγχος των αριθμών N, εάν δηλαδή αυτοί είναι πρώτοι κυρίως όταν: $(b/N) = -1$). Από το 1978 και μετά καθιερώθηκε ένας ακέραιος N

(ο οποίος ικανοποιεί την παραπάνω υπόθεση) να ορίζεται ως μία πιθανών πρώτη *Euler* βάση b.

Δύο δεκαετίες αργότερα οι μαθηματικοί Solovay και Strassen χρησιμοποίησαν την ίδια υπόθεση στα δικά τους πιθανοτικά τεστ πιστοποίησης πρώτων αριθμών. Παράλληλα με τους Solovay και Strassen ένας άλλος μαθηματικός ο Rabin χρησιμοποιώντας μια ιδιότητα απέδειξε ότι αυτή η υπόθεση δεν ικανοποιείται από κάποιον πρώτο. Ο Rabin ονόμασε την ιδιότητα αυτή $W(b)$ και την όρισε ως εξής:

1. $b^{N-1} \neq 1 \ (mod\ N)$

2. $1 < (b^m - 1,\ N)$ για κάποιο $m = (N-1)/2^k \in Z$

Αυτή η ιδιότητα αρχικά εισήχθει από τον Miller και χρησιμοποιήθηκε για την δημιουργία ενός υπο συνθήκη τεστ πιστοποίησης πρώτων.

Με λίγα λόγια εάν ο N είναι πρώτος τότε η ιδιότητα $W(b)$ δεν ικανοποιείται για κανένα b $(1 \leq b \leq N-1)$. Ο Rabin απέδειξε ότι αυτός ο αριθμός N είναι σύνθετος και μάλιστα ότι ισχύει:
$S = \{1 \leq b \leq N-1$ όπου ο N ικανοποιεί την $W(b)\}$
τότε: $\mid S \mid \geq 3/4(N-1)$ το οποίο είναι καλύτερο αποτέλεσμα από αυτό που είχε αποδώσει η τεχνική των Solovay και Strassen.

Σύμφωνα με τον μαθητή του Selfridge τον $Williams$ ο Selfridge απέδειξε ότι ο N δεν ικανοποιεί την $W(b)$ αν και μόνο αν ο N είναι ένας ισχυρής βάσης b πιθανοτικά πρώτος. Συνεπώς για μια τυχαία βάση b περιμένουμε ότι η πιθανότητα ένας σύνθετος αριθμός N να είναι πιθανοτικά πρώτος ισχυρής βάσης να είναι λιγότερη από $1/4$. Το συγκεκριμένο τεστ είναι αυτό που χρησιμοποιείται ακόμα και σήμερα από την κρυπτογραφική κοινότητα (χρησιμοποιόντας τουλάχιστον 50 τυχαίες βάσεις) για την πιστοποίηση πολύ μεγάλων πρώτων ακεραίων (για παράδειγμα η $FIPS\ 182-2,\ 2001$). Συχνά αναφέρεται ως Miller-Rabin test αλλά με την συνεισφορά του $Selfridge$ θα έπρεπε να λέγεται Miller-Rabin-Selfridge test.

Ορισμός 3.2.1 *Ο n ονομάζεται πιθανοτικά πρώτος ισχυρής βάσης b εάν ισχύει ένα από τα παρακάτω:*

- $2^s \mid n-1$
- $a^t \equiv 1(mod\ n)$ $(t = (n-1)/2)$
- $a^{t2^k} \equiv -1(mod\ n)$ *για κάποιο* k $(0 \leq k \leq s)$.

Έτσι για τυχαία επιλογή βάσης b περιμέναμε ότι η πιθανότητα ένας σύνθετος n να είναι ένας πιθανοτικά πρώτος ισχυρής βάσης b να είναι μικρότερη από $1/4$.

3.2.1 Ο Αλγόριθμος Solovay Strassen

Βασιζόμενοι στο μικρό θεώρημα του Fermat οι μαθηματικοί Solovay και Strassen έφτιαξαν τον πρώτο επίσημο αλγόριθμο πιστοποίησης πρώτων αριθμών.
Κάνοντας την παρακάτω αναδιατύπωση του μικρού θεωρήματος του Fermat:

Θεώρημα 3.2.1 *Για κάθε περιττό πρώτο αριθμό n, για κάθε αριθμό a με $0 < a < n$ ισχύει:*

$$a^{(n-1)\backslash 2} \equiv \pm 1 (mod\ n)$$

οι Solovay και Strassen απέδειξαν τα παρακάτω:

Για κάθε πρώτο αριθμό n το a είναι δευτεροβάθμιο υπόλοιπο αν και μόνο αν

$$a^{(n-1)/2} \equiv 1\ (mod\ n)$$

Το Legendre σύμβολο $(a$
$n)$ ισούται με 1 εάν το a είναι δευτεροβάθμιο υπόλοιπο *modulo* n και με -1 για κάθε πρώτο αριθμό n. Έτσι λοιπόν για κάθε πρώτο αριθμό n ισχύει:

$$(a/n) \equiv a^{(n-1)/2}\ (mod\ n)$$

Το Legendre σύμβολο μπορεί να γενικευτεί και για τους σύνθετους αριθμούς με τον παρακάτω ορισμό:

$$(a/n) = \prod (a/p_i)^{e_i}$$

με $i = 1, ..., k$ και όπου $n = \prod (p_i)^{e_i}$ με $i = 1,\ ...,\ k$ και p_i είναι πρώτος για κάθε i. Η γενίκευση αυτή ονομάζεται Jacobi symbol. Η πολυπλοκότητα του Jacobi symbol (το να υπολογίσουμε δηλαδή το a/n) είναι $O(log\ n)$. Για σύνθετο αριθμό n δεν είναι απαραίτητο ότι $(a/n) = 1$ εάν το a είναι δευτεροβάθμιο υπόλοιπο ή ότι $(a/n) = a^{(n-1)/2} (mod\ n)$. Με λίγα λόγια η απόδειξη της τελευταίας ισότητας θα μπορούσε να αποτελέσει ένα κριτήριο πιστοποίησης του n. Οι Solovay και Strassen απέδειξαν ότι αυτό επιτυγχάνεται με τυχαία επιλογή του n. Έτσι δημιουργήθηκε ο παρακάτω αλγόριθμος:

Solovay − Strassen
choose a at random from $\{1,\ 2,\ ...,\ n-1\}$
if $gcd(a,\ n) \neq 1$
then return ("composite") and stop
else
if$(a/n) \neq a^{(n-1)/2}\ (mod\ n)$
then return ("composite") and stop
else
return ("prime")

Θεώρημα 3.2.2 *Εάν ο n είναι πρώτος αριθμός τότε ο αλγόριθμος Solovay Strassen επιστρέφει "n is prime". Εάν ο n είναι περιττός σύνθετος αριθμός τότε για το λιγότερο $1/2$ απόλα τα a στο σύνολο $\{1,\ 2,\ ...,\ n-1\}$ επιστρέφει "n is composite". Ο αλγόριθμος Solovay Strassen χρησιμοποιεί*

$$O(log\ n)^3)$$

bit πράξεις.

Σάυτό το σημείο δεν είναι ξεκάθαρο ότι ο αλγόριθμος Solovay Strassen είναι αλγόριθμος πολυωνυμικού χρόνου. Ήδη γνωρίζουμε πως να υπολογίσουμε την ποσότητα $a^{(n-1)/2}\ (mod\ n)$ σε χρόνο $O((log\ n)^3)$, αλλά πως μπορούμε να υπολογίσουμε τα Jacobi symbols αποτελεσματικά; Δυστυχώς μπορούμε να υπολογίσουμε το Jacobi symbol χωρίς να παραγοντοποιήσουμε το n χρησιμοποιώντας μερικά θεωρήματα από θεωρία αριθμών με σημαντικότερο τη γενίκευση του νόμου της τετραγωνικής αμοιβαιότητας. Παρακάτω θα αριθμήσουμε μερικές από τις ιδιότητες αυτές.

1. Αν n είναι ένας θετικός περιττός ακέραιος και ισχύει: $m_1 \equiv m_2(mod\ n)$, τότε:
   $m_1/n = m_2/n$

2. Αν n είναι ένας θετικός περιττός ακέραιος τότε:

   $(2/n)= 1$, αν $n \equiv \pm 1(mod\ n)$ ή
   $\quad = -1$, αν $n \equiv \pm 3(mod\ n)$

3. Αν n είναι θετικός περιττός ακέραιος τότε:
   $((m_1 m_2)/n) = (m_1/n)(m_2/n)$
   Συγκεκριμένα εάν $m = 2^k \times t$ και t περιττός τότε:
   $(m/n) = (2/n)^k(t/n)$

4. Υποθέτουμε ότι m και n είναι περιττοί θετικοί ακέραιοι. Τότε:

   $(m/n)= -(n/m)$ εάν $m \equiv n \equiv 3(mod\ 4)$ ή
   $\quad = (n/m)$ αλλιώς.

Επομένως η πολυπλοκότητα του αλγορίθμου καθορίζεται από τον αριθμό των *modular* αναγωγών που γίνονται. Συνολικά γίνονται $O(log\ n)$ *modular* αναγωγές. Κάθε μια από τις *modular* αναγωγές που εκτελούνται χρειάζεται περίπου $O((log\ n)^2)$. Έτσι προκύπτει ότι η πολυπλοκότητα του αλγορίθμου είναι $O((log\ n)^3)$.

3.2.2 Ο Αλγόριθμος Miller-Rabin-Selfridge

Ο αλγόριθμος Miller-Rabin-Selfridge είναι ένας Monte Carlo αλγόριθμος για σύνθετους αριθμούς ο οποίος είναι επίσης γνωστός ως ο αλγόριθμος των ισχυρα ψευδοπρώτων. Ο Miller Rabin Selfridge είναι ένας καθαρά πολυωνυμικού χρόνου αλγόριθμος με πολυπλοκότητα $O((log\ n)^3)$ ακριβώς όπως ο αλγόριθμος Solovay Strassen αλλά γενικά είναι ευκολότερος στην πράξη από τον Solovay Strassen. Παρακάτω θα δείξουμε ότι ο Miller Rabin Selfridge δεν μπορεί να επιστρέψει ότι ένας αριθμός είναι σύνθετος έαν αυτός είναι πρώτος.

Ορισμός 3.2.2 *Έστω n ακέραιος. Θεωρώντας τις παρακάτω υποθέσεις δηλώνουμε το $W_n(b)$ σε έναν ακέραιο b:*

- $1 \leq b \leq n\cdot$
- $(a)\ b^{n-1} \neq 1 mod\ n$, *ή*
  $(b)\ \exists i$ *τέτοιο ώστε* $2^i|(n-1)$ *και* $1 < (b^{(n-1)/2^i} - 1,\ n) < n$

Κάθε τέτοιο ακέραιο b τον ονομάζουμε μάρτυρα της συνθετότητας του n (witness to the compositeness of n). Αυτή η υπόθεση θεωρήθηκε πρώτα από τον Miller ο οποιός την χρησιμοποίησε για να αποδώσει έναν μη πιθανοτικό αλγόριθμο εύρεσης πρώτων αριθμών χρησιμοποιώντας την εκτενή υπόθεση του Riemann. Τροποποιώντας τον αλγόριθμο του Miller και χρησιμοποιώντας την παραπάνω υπόθεση ο Rabin δημιούργησε τον παρακάτω αλγόριθμο τον οποίο ονομάζουμε αλγόριθμο Miller Rabin.

Miiller-Rabin(n, k)
$n - 1 = 2^s m$
for (i=1 to k) Do{
  Choose randomly $b_i \in 1,\ ...,\ n-1$
  Calculate $b_i^m\ mod\ n$
  $for(j = 1\ to\ s)$\{
    calculate $(b_i^m)^{2^j} mod\ m$
    if $(b_i^{n-1} \neq 1\ mod\ n)\ \longrightarrow$ return composite
    else
      if $1 < gcd(res((b_i^m)^{2^j} - 1,\ n),\ n) < n \longrightarrow$ return composite
  }
}
return prime

Θεώρημα 3.2.3 *Έστω $4 < n$ και ο n είναι σύνθετος, τότε:*
$(3(n-1)/4) \leq c(\{b|W_n(b)\})$
όπου $c(S)$ είναι ο αριθμός των στοιχείων του S.

Θεώρημα 3.2.4 *Ο παραπάνω αλγόριθμος απαιτεί για* $n-1=2^l m$ *(όπου ο* m *είναι περιττός) το πολύ*

$$k(3log_2 n + l \times log_2 n)$$

βήματα. Εάν ο n *είναι πρώτος τότε το αποτέλεσμα είναι πάντα σωστό. Για κάθε σύνθετο αριθμό* n *ο αλγόριθμος μπορει να αποφανθεί ότι ο* n *είναι πρώτος αλλα η πιθανότητα κάθε τέτοιου λάθους είναι μικρότερη από* $(1/2)^{2\times k}$

Απόδειξη:
Θάποδείξουμε πρώτα το πρωτο σκέλος του θεωρήματος. Δοσμένου αριθμού n επιλέγουμε ένα k και επιλέγουμε τυχαία ένα b_i από τα $1 \leq b_1,\ ...,\ b_k \leq n$. Έστω ότι ισχύει: $n-1=2^l m$ όπου m περιττός. Για κάποιο $b=b_i$ υπολογίζουμε το $b^m (mod\ n)$. Αυτό απαιτεί παρίπου $1,5\ log_2 m$ πολλαπλασιασμούς. Κάθε φορά λαμβάνεται ένας αριθμός $n<d$, υπολογίζεται το $d_1 res(d,n)$ και συνεχίζει. Για το λόγο αυτό παράγονται μόνο αριθμοί $a,\ b<n$ και υποδιαιρέσεις του $d<n^2$. Έπειτα υπολογίζεται το το $res\ mod\ n$ από τα $b^{2m},\ ...,\ b^{2^l m}=b^{n-1}$. Αφού ισχύει $log_2 n = l + log_2 m$ ο υπολογισμός όλων των απαραίτητων δυνάμεων του b απαιτούν $1,5\ log_2 n$ πολλαπλασιασμούς από τα $a,\ b<n$ και $1,5\ log_2 n$ υποδιαιρέσεις του $d<n^2$, για όλα μαζί δηλαδή απαιτούνται $3\ log_2 n$ βήματα. Εάν το υπόλοιπο του b^{n-1} είναι διαφορετικό από το 1 η ιδιότητα $W_n(b)$ ισχύει. Εάν είναι ίσο με 1 υπολογίζει το $(res(b^{2^m},\ n)-1,\ n)$ για κάθε $1 \leq i \leq l$. Κάθε υπολογισμός του ΜΚΔ επιτυγχάνεται κάνοντας το πολύ $log_2 n$ αφαιρέσεις και διαιρέσεις με το 2. Εάν κάποιος από αυτούς τους ΜΚΔ είναι διάφορος του 1 ή του n τότε η ιδιότητα $W_n(b)$ ισχύει και ο συνολικός αριθμός βημάτων (για k επαναλήψεις) είναι:

$$k(3log_2 n + l \times log_2 n)$$

. Εάν για κάθε $1 \leq i \leq k$ η ιδιότητα $W_n(b)$ ισχύει ο n είναι σύνθετος. Εάν για κάθε $1 \leq i \leq k$ η ιδιότητα $W_n(b)$ δεν ισχύει τότε ο n είναι πρώτος.

Όσον αφορα το δεύτερο σκέλος του θεωρήματος το λάθος αυτό μπορεί να συμβεί μόνο εάν το n που ελέγχεται είναι περιττός και τα $b_1, ..., b_k$ που επιλέγονται από ένα ιδιαίτερο "τρέξιμο" του αλγορίθμου να είναι όλα μη-μάρτυρες ($nonwitnesses$). Εξάιτίας του προηγούμενου θεωρήματος η πιθανότητα της τυχαίας επιλογή των nonwitnesses είναι μικρότερη από 1/4. Η πιθανότητα των ανεξάρτητων διαλογών των $nonwitnesses$ k είναι μικρότερη από $(1/4)^k = (1/2)^{2k}$. Επομένως η πιθανότητα για κάθε δοσμένο n, ένα ιδιαίτερο "τρέξιμο" του αλγορίθμου να παράγει μια εσφαλμένη απάντηση είναι μικρότερη από $(1/2)^{2k}$.

Πρόταση 3.2.1 *Έστω* n *ένας περιττός σύνθετος αριθμός. Εάν για κάποιον περιττό αριθμό* d *ισχύει:*

$$n-1=2^s \times d$$

τότε ο n *ονομάζεται Ισχυρά ψευδοπρώτος για τη βάση* a *εάν ισχύει:*

$$a^d \equiv 1 (mod\ n)$$

ή

$$a^{2r\times d} \equiv -1(mod\ n)$$

για κάποιο r, $\mu\epsilon$ $0 \leq r < s$. *Ισχυρά ψευδοπρώτοι είναι δηλαδή οι αριθμοί για τους οποίους το Miller-Rabin test αποτυγχάνει. Το σύνολο όλων αυτών των αριθμών* n *συμβολίζεται:* $spsp(a)$.

Θεώρημα 3.2.5 *Ο αλγόριθμος Miller Rabin είναι ένας Monte Carlo αλγόριθμος θετικού σφάλματος.*

Απόδειξη:
Θα αποδείξουμε ότι ο Miller Rabin επιστρέφει ότι ένας αριθμός n είναι σύνθετος για κάποιον πρώτο αριθμό n και περιέχει μια αντίφαση. Για να επιστρέψει ο αλγόριθμος ότι ο n είναι σύνθετος θα πρέπει να ισχύει ότι: $a^m \neq 1(mod\ n)$. Καθώς ο b υψώνεται εις το τετράγωνο σε κάθε επανάληψη μεσα στην εντολή for υπολογίζουμε όλες τις τιμές: a^m, a^{2m}, ..., $a^{2^{k-1}m}$. Καθώς ο αλγόριθμος επιστρέφει ότι ο n είναι σύνθετος υπολογίζουμε ότι:
$a^{2^i m} \neq -1(mod\ n)$ για κάθε: $0 \leq i \leq k-1$. Τώρα χρησιμοποιώντας την προυπόθεση ότι ο n είναι πρώτος από το μικρό θεώρημα του Fermat έχουμε ότι: $a^{2^k m} \equiv 1(mod\ n)$ όπου $n-1 = 2^k m$. Έτσι το $a^{2^{k-1}m}$ είναι η τετραγωνική ρίζα του 1 *modulo* n και επειδή ο n είναι πρώτος υπάρχουν μόνο 2 τετραγωνικές ρίζες του 1 *modulo* n, οι $\pm 1(mod\ n)$. Έτσι έχουμε:

$$a^{2^{k-1}m} \neq -1(mod\ n)$$

Άρα:

$$a^{2^{k-1}m} = 1(mod\ n)$$

Επομένως το $a^{2^{k-2}m}$ πρέπει να είναι τετραγωνική ρίζα του 1. Με την ίδια λογική έχουμε ότι:

$$a^{2^{k-2}m} \equiv 1(mod\ n)$$

Ακολουθώντας την ίδια διαδικασία καταλήγουμε στο

$$a^m \equiv 1(mod\ n)$$

το οποίο βέβαια αποτελεί αντίφαση αφού ο αλγόριθμος όπως έχουμε πει έχει επιστρέψει ότι ο n είναι πρώτος.

Θεώρημα 3.2.6 *Υποθέτουμε ότι ο* p *είναι ένας περιττός πρώτος αριθμός, ο* e *είναι ένας θετικός ακέραιος και ο* $gcd(a,\ p) = 1$. *Τότε η αντίφαση*

$$y^2 \equiv a(mod\ p^e)$$

δεν έχει λύσεις εάν $(a/p) = -1$ *και έχει δύο λύσεις: modulo* p^e *εάν* $(a/p) = 1$

Το παραπάνω θεώρημα μας λέει ότι η ύπαρξη τετραγωνικής ρίζας του a modulo p^e μπορεί να εξηγηθεί από τον ορισμό του Legendre symbol (a/p).

Παράλληλα με τον $Miller$ ένας άλλος μαθηματικός ο John Selfridge δούλευε έναν αλγόριθμο πιστοποίησης πρώτων αριθμών. Όπως αναφέρει ο μαθητης του ο Hugh Williams ο John Selfridge απέδειξε ότι η ιδιότητα $W(b)$ δεν ικανοποιείται από το n όπως ορίστηκε από τον Rabin. Συγκεκριμένα έαν ο n είναι πρώτος τότε ο n δεν ικανοποιεί την ιδιότητα $W(b)$ για κάθε b με $1 \leq b \leq n-1$. Ο Rabin απέδειξε ότι εάν ο n είναι σύνθετος και ισχύει: $S = 1 \leq b \leq n-1$ όπου ο n ικανοποιεί τη $W(b)$ τότε ισχύει: $\mid S \mid \geq 3/4(n-1)$.

Ο Selfridge απέδειξε ότι: η ιδιότητα $W(b)$ δεν ικανοποιείται αν και μόνο αν ο n είναι ένας πιθανοτικά πρώτος ισχυρής βάσης b.

Όπως αναφέρεται ο $Selfridge$ απέδειξε το παρακάτω θεώρημα:

Θεώρημα 3.2.7 *Έαν ο n είναι ένας πιθανοτικά πρώτος ισχυρής βάσης b τότε ο n είναι ένας Euler πιθανοτικά πρώτος για την βάση b.*

Έτσι με την εισφορά του Selfridge ο αλγόριθμος Miller-Rabin πήρε την παρακάτω μορφή και είναι ο αλγόριθμος αυτός που χρησιμοποιείται σήμερα από την κρυπτογραφική κοινότητα.

Ο Αλγόριθμος Miller-Rabin-Selfridge The Miller-Rabin-Selfridge test

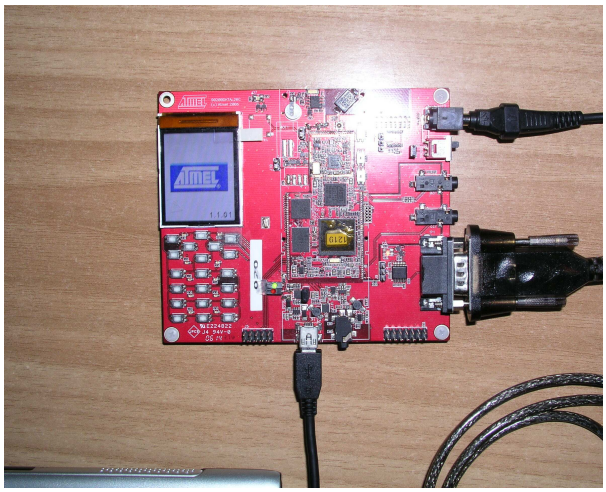
write $n-1 = 2^k m$, *m is odd*
choose randomly a such that: $1 \leq a \leq n-1$
$b \leftarrow a^m mod\ n$
$if\ b \equiv -1(mod\ n)$
  return "n is prime"
  $for\ i \leftarrow 0\ to\ k-1$
do
  $if\ b \equiv -1(mod\ n)$
  return "n is prime"
  else $b \leftarrow b^2 mod\ n$
return "n is composite"

Κεφάλαιο 4

Πιστοποίηση Πρώτων Αριθμών σε Ενσωματωμένα Συστήματα

Όπως αναφέρθηκε και στο Κεφάλαιο 1 οι πρώτοι αριθμοί αποτελούν τον θέμελιο λίθο στις περισσότερες αν όχι σε όλες τις κρυπτογραφικές εφαρμογές. Στην σημερινή εποχή, η πρόσβαση στο διαδίκτυο από οποιοδήποτε σημείο και η επικοινωνία και η εκτέλεση υπολογισμών μέσω μικρών ηλεκτρονικών συσκευών (ενσωματωμένες συσκευές) με περιορισμένο αποθηκευτικό χώρο και επεξεργαστική ισχύ έχει γίνει σχεδόν απαραίτητη. Η ανάγκη όμως για ασφαλή πρόσβαση και επικοινωνία επιβάλλει την χρήση ισχυρών κρυπταλγορίθμων λαμβάνοντας πάντα υπόψη τους περιορισμούς που εισάγει η εκάστοτε συσκευή.

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα της ανάπτυξης των αλγορίθμων πιστοποίησης πρώτων αριθμών που εξετάσθηκαν στο κεφάλαιο 3 τόσο σε φορητό υπολογιστή όσο και ένα ασύρματο VoIP(Voice over IP τηλέφωνο. Στην πρώτη ενότητα γίνεται μια σύντομη αλλά περιεκτική περιγραφή ενός ασύρματου τηλέφωνου βασισμένου στον επεξεργαστή AT76C902 της ATMEL. Στην επόμενη ενότητα παρασουσιάζεται η υλοποίηση σε γλώσσα C του αιτιοκρατικού αλγόριθμου του Miller, του πιθανοθεωρητικού αλγόριθμου των Solovay-Strassen και του αλγόριθμου των Miller, Rabin και Shelfridge καθώς και τα αποτελέσματα της εκτέλεσης των αλγορίθμων αυτών σε έναν φορητό υπολογιστή και στο ενσωματωμένο σύστημα της ATMEL AT76C902. Για την υλοποίηση αυτή χρησιμοποιήθηκε η βιβλιοθήκη αριθμητικής πολλαπλής ακρίβειας GNU-MP αναλυτική περιγραφή της οποίας δίνεται στο παράρτημα.



Σχήμα 4.1: Το ασύρματο VoIP τηλέφωνο

4.1 Η ενσωματωμένη συσκευή AT76C902

Οι συναρτήσεις που εξετάσθηκαν στην προηγούμενη ενότητα δοκιμάσθηκαν σε ένα φορητό υπολογιστή με επεξεργαστή τον T2300 με δύο πηρύνες στα 1,67GHz της Intel, μνήμη 1GB και λειτουργικό σύστημα Linux και σε ένα ασύρματο κινητό τηλέφωνο με επεξεργαστή τον AT76C902 της ATMEL το οποίο φαίνεται στο Σχήμα 4.1.

Ο επεξεργαστής AT76C902 διαθέτει δυο πηρύνες ARM: ARM7 και ARM9 ενώ δεν διαθέτει μονάδα διαχείρισης μνήμης (memory management unit- MMU). Διαθέτει μια διεπαφή εποκοινωνίας με εξωτερική μνήμη μέχρι 16MB SRAM, 16MB FLASH και 512MB SDRAM. Ο πηρύνας ARM7 υποστηρίζει το MAC layer του πρωτοκόλλου ασύρματου τοπικού δικτύου 802.11 (a/b/g) ενώ ο πηρύνας ARM9, ο οποίος διαθέτει μια μνήμη κρύπτης εντολών των 8KB και μια μνήμη κρύπτης δεδομένων των 8KB, υποστηρίζει τα πρωτόκολλα VoIP, TCP/IP, VPN καθώς και τις λειτουργίες ελέγχου τηλεφώνου.

Το εν λόγω σύστημα βασίζεται στο λειτουργικό σύστημα ucLinux ή micro-controller Linux, το οποίο είναι μια δημοφιλής παραλλαγή του κλασσικού Linux, που αναπτύχθηκε για embedded μικροεπεξεργαστές χωρίς μονάδα διαχείρισης μνήμης. Μικροεπεξεργαστές χωρίς MMU είναι πολύ συχνοί για μείωση του κόστους. Ο πυρήνας του λειτουργικού υποστηρίζει πολλές πλατφόρμες επεξεργαστών ανάμεσα στις οποίες είναι και ο ARM. Η βασική του διαφορά με το Linux πέρα της έλλειψης MMU είναι το γεγονός πως σχεδιάστηκε με σκοπό να είναι μια ιδιαίτερα συμπαγής λύση. Όπως και το Linux, έτσι και το ucLinux παρέχει ισχυρή υποστήριξη δικτυακών πρωτοκόλλων συμπεριλαμβανομένου και του TCP/IP. Επίσης υποστηρίζει διάφορα συστήματα αρχείων. Το ucLinux είναι κατά

```
 1:
 2: #include <time.h>
 3: #include <sys/time.h>
 4: #include <stdio.h>
 5: #include <gmp.h>
 6:
 7: typedef enum {FALSE,TRUE} bool;
 8:
 9: bool fermat_PrimeTest(mpz_t, int, unsigned long);
10: bool Solovay_Strassen(mpz_t, unsigned long);
11: bool Miller_Rabin_Selfridge(mpz_t, int, unsigned long);
12: bool Miller_Rabin(mpz_t, int, unsigned long);
13: bool Miller(mpz_t, double, unsigned long int);
```

Σχήμα 4.2: Το αρχείο επικεφαλίδα gmpPrimes.h

το μέγιστο δυνατό συμβατό με το κανονικό Linux και αυτό σημαίνει πως εφαρμογές που αναπτύσσονται για το Linux μπορούν με μικρές προσαρμογές να χρησιμοποιηθούν και από το ucLinux λαμβάνοντας υπόψη ενδεχομένους περιορισμούς των ενσωματωμένων συσκευών σε επεξεργαστική ισχύ και μνήμη.

Το ucLinux στο AT76C902 αξιοποιεί τον ARM7 για τις λειτουργίες του MAC layer του 802.11 πρωτοκόλλου και τον ARM9 πηρύνα για τις λειτουργίες που αναφέρθηκαν στην αρχή της ενότητας καθώς και για οποιαδήποτε άλλη εφαρμογή. Οι αλγόριθμοι πιστοποίησης πρώτων αριθμών που εξετάσθηκαν στο σύστημα αυτό έτρεξαν στον πηρύνα ARM9.

4.2 Αλγόριθμοι Πιστοποίησης Πρώτων Αριθμών σε γλώσσα C

Για κάθε αλγόριθμο που παρουσιάστηκε στο Κεφάλαιο 3 αναπτύχθηκε μια συνάρτηση σε C οι οποίες αποθηκεύτηκαν σε διαφορετικά αρχεία. Σε κάθε ένα από αυτά τα αρχεία συμπεριλήφθηκε το αρχείο επικεφαλίδα $gmpPrimes.h$ το περιεχόμενο του οποίου παρατίθεται στο Σχήμα 4.2. Συγκεριμένα, στο αρχείο αυτό περιλαμβάνονται συμπεριλήψεις των αρχείων επικεφαλίδα $gmp.h$, $stdio.h$, $sys/time.h$, δημιουργείται ο τύπος απαρίθμησης *bool* και δηλώνονται τα πρότυπα των συναρτήσεων που αναπτύχθηκαν για τους αλγορίθμους πιστοποίησης πρώτων αριθμών. Στις υποενότητες που ακολουθούν γίνεται σχολιασμός των συναρτήσεων αυτών καθώς και των αποτελεσμάτων της εκτέλεσης τους στον φορητό υπολογιστή και στο ενσωματωμένο σύστημα της ATMEL AT76C902 που περιγράφηκαν στην προηγούμενη ενότητα.

$\#bit$	φορητός
128	$0,027sec$
256	$0,31sec$
512	$5,1sec$

Πίνακας 4.1: Μέσος χρόνος εκτέλεσης της συνάρτησης Miller (t_M) στον φορητό υπολογιστή.

4.2.1 Ο αιτιοκρατικός αλγόριθμος του Miller

Στο Σχήμα 4.3 παρουσιάζεται η συνάρτηση σε C η οποία υλοποιεί τον αιτιοκρατικό αλγόριθμο πιστοποίησης πρώτων αριθμών του Miller. Στην αρχή συμπεριλαμβάνεται το αρχείο επικεφαλίδα $gmpPrimes.h$ και δηλώνονται οι δυαδικές μεταβλητές $PRIME$, η οποία σε κάθε χρονική στιγμή εκτέλεσης της συνάρτησης κρατάει το αποτέλεσμα (δηλαδή αν ο υπο εξέταση αριθμός είναι πρώτος ή σύνθετος) και η $FOUND$ η οποία χρησιμοποιείται σε διάφορα σημεία του αλγόριθμου ως μεταβλητή τερματισμού.

Στη συνέχεια δηλώνονται όλες οι ακέραιες μεταβλητές πολλαπλής ακρίβειας με χρήση του τύπου mpz_t, οι μεταβλητές κινητής υποδιαστολής πολλαπλής ακρίβειας με τον τύπο mpf_t και οι μη προσημασμένοι ακέραιοι (unsigned long int).

Στις γραμμές $10-12$ γίνεται αρχικοποίηση των μεταβλητών με την δήλωση mpz_t init για τους ακέραιους πολλαπλής ακρίβειας και mpf_t init για τους αριθμούς κινητής υποδιαστολής πολλαπλής ακρίβειας [5]. Εάν ο n είναι τέλεια δύναμη τότε η μεταβλητή $PRIME$ παίρνει την τιμή $FALSE$ σε αντίθετη περίπτωση υπολογίζεται η παράσταση $n-1=2^s m$ και για m περιττό $s=s+1$ (γραμμές $16-21$). Στη συνέχεια υπολογίζεται $f=min(ceil(k*log^2 n, n)$ (γραμμές $33-27$). Εάν ισχύει $f>n$ η τιμή του n τοποθετείται στο f και όσο $f\geq 2$ και η μεταβλητή $PRIME$ έχει την τιμή $TRUE$ (γραμμή 30) γίνεται έλεγχος εάν ο b διαιρεί το n. Αν αυτό ισχύει η μεταβλητή $PRIME$ παίρνει την τιμή $FALSE$ (γραμμή 31) αλλιώς υπολογίζεται η δύναμη $a=b^{n-1}\ (mod\ n)$, και αν αυτή είναι διάφορη του 1 τότε η μεταβλητή $PRIME$ παίρνει την τιμή $FALSE$. Σε περίπτωση που είναι ίση με 1 τότε υπολογίζεται: $a=b^m\ (mod\ n)$ (γραμμή 36). Στη συνέχεια γίνεται σύγκριση του a με το r και το rop και αν ισχύει: $a\neq r$ και $a\neq rop$ τότε η μεταβλητή $found$ παίρνει την τιμή $FALSE$ (γραμμή 38) και εκτελούνται οι παρακάτω ενέργειες: Υπολογίζεται το $max\ j<s:(a^m)^{2^j}\neq 1(mod\ n)$ και αν ισχύει η μεταβλητή $found$ παίρνει την τιμή $TRUE$ (γραμμές $39-44$). Αλλιώς για $j=j-1$ η μετβλητή p παίρνει την τιμή $p/2$ και επαναλαμβάνεται η παραπάνω διαδικασία. Στη γραμμή 50 γίνεται σύγκριση των μεταβλητών q, rop και αν είναι διαφορετικές τότε η μεταβλητή $prime$ παίρνει την τιμή $FALSE$. Με την δήλωση $mpz\_nextprime$ [5] τοποθετείται στην τιμή της μεταβλητής b η τιμή του αμέσως επόμενου πρώτου. Τέλος γίνεται αποδέσμευση του χώρου μνήμης όλων των μεταβλητών που χρησιμοποιήθηκαν στον αλγόριθμο και επιστρέφεται με την εντολή $return$, κάθε φορά η τιμή της μεταβλητής $PRIME$ (γραμμή 59).

```
#include "gmpPrimes.h"

bool Miller(mpz_t n, double k, unsigned long int logn)
{
  bool prime=TRUE, found=FALSE;
  mpz_t r, rm, rop, m, f, b, a, p, q;
  mpf_t u, v;
  unsigned long int s=1, j=1;

  mpz_init(m);mpz_init(b);mpz_init(rm);mpz_init(a);
  mpz_init(rop);mpz_init(f);mpz_init(p);mpz_init(q);
  mpf_init(u);mpf_init(v);mpz_init_set_ui(r, 1);mpz_init_set_si(rm, -1);

  if (mpz_perfect_power_p(n)!=0) prime = FALSE;
  else {
   mpz_sub(rop, n, r);
   while(found==FALSE){
        mpz_ui_pow_ui(b, 2, s);
        mpz_divexact(m, rop, b);
        if (mpz_odd_p(m)==0) s++;
        else found = TRUE;
   }
   mpz_ui_pow_ui(b, logn, 2);
   mpf_set_z(u, b);mpf_set_d(v, k);
   mpf_mul(v, u, v);
   mpf_ceil(u, v);
   mpz_set_f(f, u);
   if (mpz_cmp(f, n)>0) mpz_set(f,n);
   mpz_set_ui(b, 2);
   while((mpz_cmp(b,f)<=0) && prime==TRUE){
        if(mpz_divisible_p(n, b)!=0) prime==FALSE;
        else{
             mpz_powm(a, b, rop, n);
             if (mpz_cmp(a, r)!=0) prime = FALSE;
             else{
                  mpz_powm(a, b, m, n);
                  if ((mpz_cmp(a, r)!=0) && (mpz_cmp(a, rop)!=0)){
                       found = FALSE;
                       j=s-1;
                       mpz_ui_pow_ui(p, 2, j);
                       mpz_mul(p, p, m);
                       while((found==FALSE) && (j>=1)){
                            mpz_powm(q, b, p, n);
                            if (mpz_cmp(q, r)!=0) found = TRUE;
                            else {
                                 j--;
                                 mpz_divexact_ui(p, p, 2);
                            }
                       }
                       if (mpz_cmp(q, rop)!=0) prime=FALSE;
                  }
             }
        }
        mpz_nextprime(b, b);
   }
  }
  mpz_clear(m);mpz_clear(b);mpz_clear(rm);mpz_clear(a);mpz_clear(rop);
  mpz_clear(r);mpz_clear(f);mpz_clear(p);mpz_clear(q);mpf_clear(u);mpf_clear(v);
  return prime;
}
```

Σχήμα 4.3: Ο αλγόριθμος του Miller

Η απόδοση της συνάρτησης αυτής εξετάσθηκε στον φορητό υπολογιστή που αναφέρθηκε στην αρχή της ενότητας. Συγκεκριμένα, το πείραμα που έλαβε χώρα περιλάμβανε αρχικά την δημιουργία μιας ψευδοτυχαίας ακολουθίας μεγάλων ακεραίων μήκους 10000. Κάθε ακέραιος της ακολουθίας ήταν $l-bits$ όπου $l = 128, 256,$ και 512. Για κάθε έναν από τους ακεραίους της ακολουθίας καλούνταν η παραπάνω συνάρτηση. Πριν την κλήση και μετά την κλήση της συνάρτησης καταχωρούνταν η ώρα του συστήματος και στο τέλος υπολογίζονταν η διαφορά των δύο χρόνων προκειμένου να υπολογισθεί ο χρόνος που χρειάστηκε η συνάρτηση για να αποκριθεί αν ο αριθμός της ακολουθίας είναι πρώτος ή σύνθετος (elapsed time). Στην συνέχεια, οι επιμέρους χρόνοι αθροίζονταν και με την ολοκλήρωση του πειράματος το άθροισμα διαιρούνταν με το πλήθος των ακεραίων που εξετάσθηκαν προκειμένου να υπολογισθεί ο μέσος χρόνος που κάνει ο αιτιοκρτικός αλγόριθμος πιστοποίησης πρώτων αριθμών του Solovay.

Στον Πίνακα 4.1 παρατίθενται οι χρόνοι που υπολογίσθηκαν κατά την εκτέλεση του παραπάνω πειράματος. Για την συνάρτηση f που καθορίζει το πλήθος των βημάτων εκτέλεσης του αλγόριθμου του Miller επιλέχθηκε για την μεταβλητή k η τιμή 10 καθώς δίνει ένα καλό πάνω φράγμα για την επιλογή πρώτων αριθμών μικρότερους από την f. Όπως φαίνεται και στον πίνακα, ο χρόνος εκτέλεσης του αλγόριθμου είναι για κάθε ακέραιο της τάξης του δευτερολέπτου. Είναι δε ικανοποιητικός μόνο για την περίπτωση μικρών ακεραίων δηλαδή για $l = 128$ και 256. Για μεγάλους ακεραίους, $l = 512$ ο χρόνος εκτέλεσης είναι περίπου $5sec$ και σχεδον εξαπλάσιος για ενσωματωμένη συσκευή κάτι που τον καθιστά ασύμφορο για εφαρμογές πραγματικού χρόνου που απαιτούν άμεσες αποκρισεις.

4.2.2 Ο Πιθανοτικός Αλγόριθμος των Solovay-Strassen

Στο Σχήμα 4.4 παρουσιάζεται η συνάρτηση σε C η οποία υλοποιεί τον αλγόριθμο πιστοποίησης πρώτων αριθμών των Solovay και Strassen. Στην αρχή συμπεριλαμβάνεται το αρχείο επικεφαλίδα $gmpPrimes.h$ και δηλώνονται οι μεταβλητές a, d, jac, expon, q, r, one ως ακέραιοι πολλαπλής ακρίβειας με χρήση του τύπου mpz_t.

Εν συνεχεία ορίζεται η μεταβλητή κατάσταση της γεννήτριας ψευδοτυχαίων αριθμών. Στις γραμμές $10 - 12$ γίνεται αρχικοποίηση των μεταβλητών και για την ακολουθία ψευδοτυχαίων αριθμών μπαίνει σαν αρχική συνθήκη η τιμή του *seed*. Όσο ισχύει $a \neq 0$ γίνεται η δημιουργία ψευδοτυχαίων αριθμών οι οποίοι κυμαίνονται από 1 μέχρι $n - 1$. Στη γραμμή 18 υπολογίζουμε τον ΜΚΔ των $(a,\ n)$ και η τιμή που προκύπτει τοποθετείται στη μεταβλητή d, και αν $d \neq 1$ τότε η μεταβλητή $PRIME$ παίρνει την τιμή $FALSE$. Σε αντίθετη περίπτωση υπολογίζονται τα παρακάτω: $jac = a/n$, $jac = jac\ (mod\ n)$, $expon = n - one$, $q = expon/2$ και $r = a^q\ (mod\ n)$. (Έχουμε δηλαδή υπολογίσει το $r = a^{(n-1)/2}\ (mod\ n)$). Αν ισχύει $jac \neq r$ τότε η μεταβλητή $PRIME$ παίρνει την τιμή $FALSE$ σε αντίθετη περίπτωση την τιμή $TRUE$. Τέλος (γραμμές $30 - 31$) γίνεται αποδέσμευση του χώρου μνήμης που καταλάμβαναν οι μεταβλητές που χρησιμοποιήθηκαν στον αλγόριθμο και στη γραμμή 32 ο αλγόριθμος επιστρέφει

```
 1: #include "gmpPrimes.h"
 2:
 3: bool Solovay_Strassen(mpz_t n, unsigned long seed)
 4: {
 5:   mpz_t a, d, jac;
 6:   gmp_randstate_t state;
 7:   mpz_t expon, q, r, one;
 8:   bool prime;
 9:
10:   mpz_init(a);mpz_init(d);mpz_init(q);
11:   mpz_init(expon);mpz_init(jac);mpz_init(r);
12:   mpz_init_set_ui(one, 1);
13:
14:   gmp_randinit_mt(state);
15:   gmp_randseed_ui(state, seed);
16:   while (mpz_cmp_ui(a, 0)==0) mpz_urandomm(a, state, n);
17:
18:   mpz_gcd(d, a, n);
19:   if (mpz_cmp_ui(d, 1)!=0) prime=FALSE;
20:   else{
21:    mpz_set_si(jac, mpz_jacobi(a, n));
22:    mpz_mod(jac, jac, n);
23:    mpz_sub(expon, n, one);
24:    mpz_divexact_ui(q, expon, 2);
25:    mpz_powm(r, a, q, n);
26:    if (mpz_cmp(jac, r)!=0) prime=FALSE;
27:    else prime=TRUE;
28:   }
29:
30:   mpz_clear(q);mpz_clear(expon);mpz_clear(a);
31:   mpz_clear(d);mpz_clear(r);mpz_clear(one);gmp_randclear(state);
32:   return prime;
33: }
34:
```

Σχήμα 4.4: Ο αλγόριθμος Solovay-Strassen

t_{SS}		
$\#bit$	φορητός	AT76C902
128	$0,18ms$	$11,3ms$
256	$0,97ms$	$36,4ms$
512	$6,31ms$	$0,17sec$
1024	$34,2ms$	$0,73sec$
2048	$0,24sec$	$4,67sec$

Πίνακας 4.2: Μέσος χρόνος εκτέλεσης της συνάρτησης Solovay-Strassen (t_{SS}) στον φορητό υπολογιστή και στην ενσωματωμένη συσκευή AT76C902.

την τιμή του $PRIME$.

Η απόδοση της συνάρτησης αυτής εξετάσθηκε τόσο στον φορητό υπολογιστή που αναφέρθηκε στην αρχή της ενότητας όσο και στο ασύρματο VoIP τηλέφωνο που περιγράφηκε στην υποενότητα 4.1. Συγκεκριμένα, το πείραμα που έλαβε χώρα περιλάμβανε αρχικά την δημιουργία μιας ψευδοτυχαίας ακολουθίας μεγάλων ακεραίων μήκους 10000. Κάθε ακέραιος της ακολουθίας ήταν $l - bits$ όπου $l = 128, 256, 512, 1024$ και 2048. Για κάθε έναν από τους ακεραίους της ακολουθίας καλούνταν η παραπάνω συνάρτηση. Πριν την κλήση και μετά την κλήση της συνάρτησης καταχωρούνταν η ώρα του συστήματος και στο τέλος υπολογίζονταν η διαφορά των δύο χρόνων προκειμένου να υπολογισθεί ο χρόνος που χρειάστηκε η συνάρτηση για να αποκριθεί αν ο αριθμός της ακολουθίας είναι πρώτος ή σύνθετος (elapsed time). Στην συνέχεια, οι επιμέρους χρόνοι αθροίζονταν και με την ολοκλήρωση του πειράματος το άθροισμα διαιρούνταν με το πλήθος των ακεραίων που εξετάσθηκαν προκειμένου να υπολογισθεί ο μέσος χρόνος που κάνει ο αλγόριθμος πιστοποίησης πρώτων αριθμών των Solovay και Strassen.

Στον Πίνακα 4.2 παρατίθενται οι χρόνοι που υπολογίσθηκαν κατά την εκτέλεση του παραπάνω πειράματος. Στην πρώτη στήλη εμφανίζονται οι χρόνοι που ελήφθησαν κατά την εκτέλεση του πειράματος στον φορητό υπολογιστη και στη δεύτερη στήλη εμφανίζονται οι χρόνοι που ελήφθησαν κατά την εκτέλεση του πειράματος στο ασύρματο τηλέφωνο. Παρατηρούμε ότι οι χρόνοι στον φορητό υπολογιστή ειναι της τάξης του ms για ακεραίους μήκους $1024 - bit$ ενώ για μεγαλύτερους φτάνουμε στα $0,24sec$. Στην περίπτωση που ασύρματου τηλεφώνου τα πράγματα είναι διαφορετικά. Οι χρόνοι είναι της ταξης του ms για ακεραίους μήκους $256 - bit$ ενώ από εκεί και πάνω οι χρόνοι είναι της τάξης του δευτερολέπτου με μέγιστη τιμή όπως άλλωστε αναμένονταν τα $4,67sec$ για ακεραίους μήκους $2048 - bit$. Να τονισθεί ότι οι χρόνοι που ελήφθησαν κατά την εκτέλεση του πειράματος στο ασύρματο τηλέφωνο ήταν κάτω από απόλυτα ιδανικές συνθήκες υπό την έννοια ότι μόνο η συνάρτηση πιστοποίησης ήταν σε κατάσταση εκτέλεσης ενώ στον φορητό υπήρχαν και άλλες εφαρμογές σε κατάσταση εκτέλεσης.

4.2.3 Ο Πιθανοτικός Αλγόριθμος των Miller-Rabin-Selfridge

Στο Σχήμα 4.5 παρουσιάζεται η συνάρτηση σε C η οποία υλοποιεί τον αλγόριθμο πιστοποίησης πρώτων αριθμών των Miller, Rabin και Shelfridge. Στην αρχή συμπεριλαμβάνεται το αρχείο επικεφαλίδα $gmpPrimes.h$ και δηλώνονται οι μεταβλητές a, m, rop, b, r, p, q ως ακέραιοι πολλαπλής ακρίβειας με τη χρηση του τύπου mpz_t και οι $i, s = 1, j$ ως μη προσημασμένοι ακέραιοι. Επίσης, δηλώνονται οι δυαδικές μεταβλητές $PRIME$, η οποία σε κάθε χρονική στιγμή εκτέλεσης της συνάρτησης κρατάει το αποτέλεσμα (δηλαδή αν ο υπο εξέταση αριθμός είναι πρώτος ή σύνθετος) και η $FOUND$ η οποία χρησιμοποιείται σε διάφορα σημεία του αλγόριθμου ως μεταβλητή τερματισμού.

Με χρήση της δήλωσης gmp_randstate_t state ορίζεται η μεταβλητή κατάσταση της γεννήτριας ψευδοτυχαίων αριθμών. Στις γραμμές $10-11$ αρχικοποιούνται όλες οι μεταβλητές. Αμέσως μετά στην ακολουθία ψευδοτυχαίων αριθμών που δημιουργείται μπαίνει σαν αρχική συνθήκη η τιμή του *seed*. Στη συνέχεια υπολογίζονται: $rop = n - r$ (γραμμή 14) και όσο ισχύει: $FOUND = FALSE$ υπολογίζονται: $b = 2^s$, $m = rop/b$ και για m μη μηδενικό και περιττό η διαδικασία επαναλαμβάνεται με $s = s + 1$. Σε αντίθετη περίπτωση $FOUND = TRUE$. Όσο ισχύει $PRIME = FALSE$, $i \leq k$ και $a \neq 0$ γίνεται η δημιουργία ψευδοτυχαίων αριθμών οι οποίοι κυμαίνονται από 1 μέχρι $n - 1$. Στη συνέχεια υπολογίζεται ο ΜΚΔ των $(a,\ n)$ και η τιμή του τοποθετείται στη μεταβλητή p (γραμμή 26). Εάν ισχύει $p \neq r$ τότε η μεταβλητή $found$ παίρνει την τιμή $TRUE$ και η $prime$ την τιμή $FALSE$. Σε αντίθετη περίπτωση (γραμμή 31) υπολογίζεται: $p = a^m\ (mod\ n)$ και σε περίπτωση που $p = 1$ ή $p = rop$ η μεταβλητή $PRIME$ παίρνει την τιμή $TRUE$ αλλιώς γίνεται ανάθεση στο j της τιμής 1 και στο q της τιμής του m (γραμμές $34-35$)(η μεταβλητή $PRIME$ συνεχίζει να έχει την τιμή $FALSE$). Όσο $j \leq s$ και η μεταβλητή $STOP$ έχει την τιμή $FALSE$ γίνεται σύγκριση των μεταβλητών r και rop και αν αυτές είναι ίσες τότε στη μεταβλητή $STOP$ ανατίθεται η τιμή $TRUE$ (γραμμές $39-40$) αλλιώς η μεταβλητή j αυξάνεται κατά μία μονάδα και υπολογίζονται οι ποσότητες: $q = q \times 2$ και $p = a^q (mod\ n)$ (γραμμές $43-44$). Η διαδικασία επαναλαμβάνεται. Εάν η μεταβλητή $STOP$ πάρει την τιμή $FALSE$ τότε οι μεταβλητές $prime$ και $found$ παίρνουν τις τιμές $FALSE$ και $TRUE$ αντίστοιχα (γραμμές $47-48$). Τέλος, γίνεται αοδέσμευση του χώρου μνήμης από τις μεταβλητές που χρησιμοποιήθηκαν στον αλγόριθμο (γραμμές $54-55$) . Ο αλγόριθμος επιστρέφει κάθε φορά την τιμή της μεταβλητής $PRIME$.

Η απόδοση της συνάρτησης αυτής εξετάσθηκε τόσο στον φορητό υπολογιστή που αναφέρθηκε στην αρχή της ενότητας όσο και στο ασύρματο VoIP τηλέφωνο. Συγκεκριμένα, το πείραμα αποτελούνταν από την δημιουργία μιας ψευδοτυχαίας ακολουθίας μεγάλων ακεραίων μήκους 10000. Κάθε ακέραιος της ακολουθίας ήταν $l - bits$ ο καθένας όπου $l = 128, 256, 512, 1024$ και 2048. Για κάθε έναν από τους ακεραιους της ακολουθίας καλούνταν η παραπάνω συνάρτηση. Πριν την κλήση και μετά την κλήση της συνάρτησης καταχωρούνταν η ώρα του συστήματος και στο τέλος υπολογίζονταν η διαφορά των δύο χρόνων προκειμένου να υπολο-

```
1:  #include "gmpPrimes.h"
2:
3:  bool Miller_Rabin_Selfridge(mpz_t n, int k, unsigned long seed)
4:  {
5:    mpz_t a, m, rop, r, b, p, q;
6:    gmp_randstate_t state;
7:    unsigned long int i, s=1, j;
8:    bool prime=TRUE, found=FALSE, stop;
9:
10:   mpz_init(a);mpz_init(m);mpz_init(q);mpz_init(b);
11:   mpz_init(rop);mpz_init(p);mpz_init_set_ui(r, 1);
12:   gmp_randinit_mt(state);gmp_randseed_ui(state, seed);
13:
14:   mpz_sub(rop, n, r);
15:   while(found==FALSE){
16:     mpz_ui_pow_ui(b, 2, s);
17:     mpz_divexact(m, rop, b);
18:     if (mpz_even_p(m)!=0) s++;
19:     else found = TRUE;
20:   }
21:
22:   i=0;
23:   found=FALSE;
24:   while((found==FALSE) && (i<k)){
25:    while (mpz_cmp_ui(a, 0)==0)mpz_urandomm(a, state, n);
26:    mpz_gcd(p, a, n);
27:    if(mpz_cmp(p, r)!=0){
28:          found=TRUE;prime=FALSE;
29:    }
30:    else{
31:          mpz_powm(p, a, m, n);
32:          if ((mpz_cmp(p, r)==0) || (mpz_cmp(p, rop)==0)) found=TRUE;
33:          else{
34:                j=1;
35:                mpz_set(q,m);
36:                mpz_mul_ui(q, q, 2);
37:                mpz_powm(p, a, q, n);
38:                stop=FALSE;
39:                while((stop==FALSE) && (j<s)){
40:                      if (mpz_cmp(p, rop)==0) stop=TRUE;
41:                      else {
42:                            j++;
43:                            mpz_mul_ui(q, q, 2);
44:                            mpz_powm(p, a, q, n);
45:                      }
46:                }
47:                if(stop==FALSE){
48:                      prime=FALSE;found=TRUE;
49:                }
50:          }
51:    }
52:    i++;
53:   }
54:   mpz_clear(a);mpz_clear(m);mpz_clear(rop);mpz_clear(b);
55:   mpz_clear(r);mpz_clear(p);gmp_randclear(state);mpz_clear(q);
56:   return prime;
57: }
```

Σχήμα 4.5: Ο αλγόριθμος Miller-Rabin-Selfridge

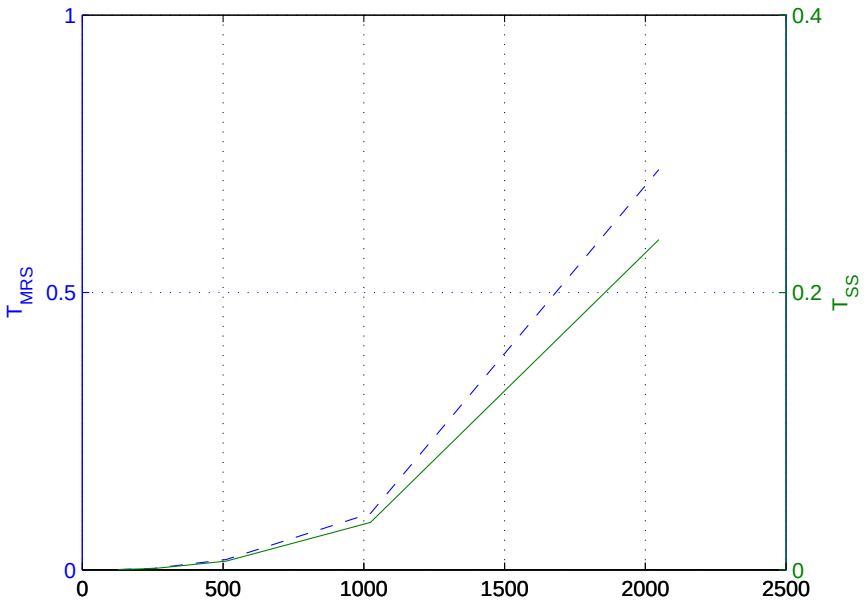
t_{MRS}		
$\#bit$	φορητός	AT76C902
128	$0,497ms$	$24,8ms$
256	$2,65ms$	$92,3ms$
512	$18,71ms$	$0,47sec$
1024	$0,101sec$	$2,11sec$
2048	$0,72sec$	$14,07sec$

Πίνακας 4.3: Μέσος χρόνος εκτέλεσης της συνάρτησης Miller-Rabin-Shelfridge (t_{MRS}) στον φορητό υπολογιστή και στην ενσωματωμένη συσκευή AT76C902.

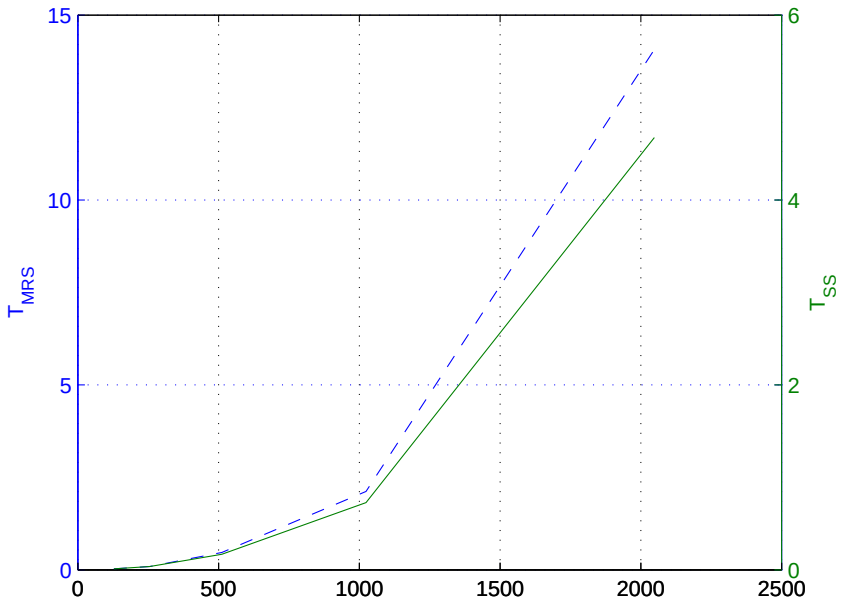
γισθεί ο χρόνος που χρειάζεται η συνάρτηση κάθε φορά για να απαντήσει αν ο αριθμός της ακολουθίας είναι πρώτος ή σύνθετος (elapsed time). Στην συνέχεια, οι επιμέρους χρόνοι αθροίζονταν και με την ολοκλήρωση του πειράματος το άθροισμα διαιρούνταν με το πλήθος των ακεραίων που εξετάσθηκαν προκειμένου να υπολογισθεί ο μέσος χρόνος που κάνει ο αλγόριθμος πιστοποίησης πρώτων αριθμών των Miller, Rabin και Selfridge.

Στον Πίνακα 4.3 παρατίθενται οι χρόνοι που υπολογίσθηκαν κατά την εκτέλεση του παραπάνω πειράματος. Στην πρώτη στήλη εμφανίζονται οι χρόνοι που ελήφθησαν κατά την εκτέλεση του πειράματος στον φορητό υπολογιστή και στη δεύτερη στήλη εμφανίζονται οι χρόνοι που ελήφθησαν κατά την εκτέλεση του πειράματος στο ασύρματο τηλέφωνο. Παρατηρούμε ότι οι χρόνοι στον φορητό υπολογιστή ειναι της τάξης του ms για ακεραίους μήκους μέχρι και $512 - bit$ ενώ για μεγαλύτερους φτάνουμε στα $0,72sec$ για ακεραίους $2048 - bit$. Για την περίπτωση του ασύρματου τηλεφώνου τα πράγματα είναι διαφορετικά. Οι χρόνοι είναι της ταξης του ms για ακεραίους μήκους μέχρι και $256 - bit$ ενώ από εκεί και πάνω οι χρόνοι είναι της τάξης του δευτερολέπτου με μέγιστη τιμή όπως άλλωστε αναμένονταν τα $14,07$ δευτερόλεπτα για ακεραίους μήκους $2048 - bit$. Να τονισθεί ότι οι χρόνοι που ελήφθησαν κατά την εκτέλεση του πειράματος στο ασύρματο τηλέφωνο ήταν κάτω από απόλυτα ιδανικές συνθήκες υπό την έννοια ότι μόνο η συνάρτηση πιστοποίησης πρώτων αριθμών ήταν σε κατάσταση εκτέλεσης ενώ στον φορητό υπολογιστή υπήρχαν και άλλες εφαρμογές σε κατάσταση εκτέλεσης.

Όπως αναλύθηκε και στο κεφάλαιο 3, στον αλγόριθμο των Miller-Rabin-Shelfridge επιχειρείται η ανεύρεση ενός μάρτυρα συνθετότητας. Δοκιμάζονται κάθε φορά k υποψήφιοι ακέραιοι οι οποίοι επιλέγονται κατά τρόπο τυχαίο. Η μεταβλητή k καθορίζει κατα τρόπο σημαντικό την απόδοση του αλγορίθμου όπως είδαμε και στην θεωρητική ανάλυση που προηγήθηκε και η τιμή εξαρτάται ισχυρά από την ποιότητα της γεννήτριας τυχαίων αριθμών που θα χρησιμοποιήσουμε. Δηλαδή, μια καλή γεννήτρια τυχαίων αριθμών περιορίζει σημαντικά την τιμή της μεταβλητής k με μοναδική επιβάρυνση αυτή της γέννησης των τυχαίων αριθμών, ενώ μια μέτρια άλλα γρήγορη γεννήτρια τυχαίων αριθμών μας οδηγεί στην επιλογή μεγαλύτερης τιμής για το k. Στο πείραμα μας επιλέξαμε για γεννήτρια τον Linear Congruential Generator που ειναι γρήγορη γεννήτρια αλλά με μέτριες ιδιότητες τυχαιότητας και επιλέξαμε για τιμή της k το 10.



Σχήμα 4.6: Γραφική Παράσταση των χρόνων t_{SS} και t_{MRS} που ελήφθησαν κατά την εκτέλεση του πειράματος στον φορητό υπολογιστή



Σχήμα 4.7: Γραφική Παράσταση των χρόνων t_{SS} και t_{MRS} που ελήφθησαν κατά την εκτέλεση του πειράματος στον ασύρματο $VoIP$ τηλέφωνο

Στα Σχήματα 4.6 και 4.7 παρουσιάζονται οι γραφικές παραστάσεις των μέσων χρόνων εκτέλεσης των αλγορίθμων Solovay-Strassen και Miller-Rabin-Shelfridge συναρτήσει του μήκους l σε *bit* των προς εξέταση ακεραίων. Στο Σχήμα 4.6 παρουσιάζονται οι χρόνοι εκτέλεσης των αλγορίθμων στο notebookκαι στο Σχήμα 4.7 παρουσιάζονται οι χρόνοι εκτέλεσης στο AT76C902. Και στα δύο διαγράμματα διακρίνουμε ότι για μήκη ακεραίων $l \leq 512$ η απόδοση και των δύο αλγορίθμων ελάχιστα διαφέρουν ενώ για μήκη ακεραίων $l > 512$ ο μέσος χρόνος εκτέλεσης του αλγόριθμου Miller-Rabin-Shelfridge είναι μεγαλύτερος αλλά με μικρότερη πιθανότητα να κάνει λάθος όπως αναφέραμε και στο κεφάλαιο 3.

Κεφάλαιο 5

Η βιβλιοθήκη αριθμητικής πολλαπλής ακρίβειας GMP

Ορισμός 5.0.1 *Η GMP: The Gnu Multiple Precision arithmetic library είναι μια βιβλιοθήκη γραμμένη σε γλώσσα C, για αριθμητική ακεραίων, ρητών και αριθμών κινητής υποδιαστολής αυθαίρετης ακρίβειας. Σκοπός της είναι η γρηγορότερη δυνατή αριθμητική για κάθε πρόβλημα το οποίο χρειάζεται μεγάλης ακρίβειας αριθμούς και οι οποίοι υποστηρίζονται από τους βασικούς τύπους της C.*

Πολλά προβλήματα χρησιμοποιούν αριθμούς ακρίβειας μερικών εκατοντάδων *bit*, όμως υπάρχουν προβλήματα τα οποία χρησιμοποιούν αριθμούς ακρίβειας χιλιάδων (και σε μερικές περιπτώσεις εκατομμυρίων) *bit* ακρίβειας. Η GMP χρησιμοποιώντας κατάλληλους αλγορίθμους υποστηρίζει και τις δύο περιπτώσεις προβλημάτων.

5.1 Τύποι και Συναρτήσεις της GMP

Για να χρησιμοποιηθεί η GMP σε μια εφαρμογή απαιτείται η συμπερίληψη του αρχείου επικεφαλίδα gmp.h με τη χρήση της εντολής του προεπεξεργαστή *#include*. Στο αρχείο αυτό έχουν συλλεχθεί όλες οι δηλώσεις που χρειάζεται μια εφαρμογή για την χρήση της GMP.

Για τις συναρτήσεις της GMP με παράμετρο δείκτη σε αρχείο που περιλαμβάνονται και στο αρχείο επικεφαλίδα *stdio.h* γίνεται συμπερίληψη του *stdio.h*.

Στην GMP υπάρχουν τρείς κύριοι τύποι αριθμών:

- Προσημασμένοι ακέραιοι πολλαπλής ακρίβειας οι οποίοι εκπροσωπούνται από

τον τύπο *mpz_t*. Υπάρχουν περίπου 150 συναρτήσεις για προσημασμένους ακεραίους.

- Ρητοί αριθμοί (Rational numbers) πολλαπλής ακρίβειας οι οποίοι εκπροσωπούνται από τον τύπο *mpq_t* Υπάρχουν περίπου 40 συναρτήσεις με παραμέτρους ρητούς.
- Αριθμοί κινητής υποδιαστολής (floating point numbers)που εκπροσωπούνται από τον τύπο *mpf_t*. Οι αριθμοί κινητής υποδιαστολής είναι ένας συνδυασμός της ακρίβειας του σημαντικού μέρους και της περιορισμένης ακρίβειας του εκθέτη. Σ′ αυτή την κατηγορία υπάρχουν περίπου 60 συναρτήσεις.

Τέλος, η *GMP* διαθέτει συναρτήσεις γέννησης τυχαίων αριθμών αυθαιρετης ακρίβειας.

5.2 Συναρτήσεις της *GMP* για την υλοποίηση των αλγορίθμων πιστοποίησης πρώτων αριθμών

5.2.1 Συναρτήσεις Ακεραίων Αριθμών

1. Εισαγωγικές συναρτήσεις και συναρτήσεις ανάθεσης
   - void *mpz_init*(*mpz_t* integer) Εισαγωγή ακεραίου και τοποθέτηση της τιμής του στο 0.
   - void *mpz_init_set_ui*(*mpz_t* rop, unsigned long int op) Εισαγωγή ακεραίου και ανάθεση στην τιμή του την τιμή ενός μη προσημασμενου ακεραίου.
   - void *mpz_clear*(*mpz_t* integer) Ελευθέρωση του διαστήματος που καταλαμβάνεται από τον ακέραιο. Η συνάρτηση αυτή καλείται για όλες τις μεταβλητές *mpz_t* τις οποίες δεν τις χρησιμοποιούμε ξανά.
   - void *mpz_set_si*(*mpz_t* rop, signed long int op)
     Ανάθεση της τιμής ενός προσημασμένου ακεραίου στην τιμή ενός ήδη εισαχθέν ακεραίου.
2. Αριθμητικές συναρτήσεις
   - void *mpz_sub*(*mpz_t* rop, *mpz_t op*_1, *mpz_top*$_2$ Τοποθέτηση στο *rop* τη διαφορά $op_1 - op_2$.
   - void *mpz_mul*(*mpz_t* rop, *mpz_t* op_1, *mpz_top*$_2$ Τοποθέτηση στο *rop* το γινόμενο $op_1 \times op_2$.
3. Συναρτήσεις διαιρέσεων

- void *mpz_divexact*(*mpz_t* r, *mpz_t* n, *mpz_t* d) Τοποθέτηση στο q το $n \setminus d$. Αυτή η συνάρτηση παράγει σωστό αποτέλεσμα μόνο εάν είναι γνωστό ότι το d διαιρεί το n.
- int *mpz_divisible_p*(*mpz_t* n, *mpz_t* d) Επιστρέφει μη μηδενικό αποτέλεσμα εάν το n διαιρείται ακριβώς απο το d
- void *mpz_mod*(*mpz_t* r, *mpz_t* n, *mpz_t* d) Τοποθέτηση στο r το n $(mod\ d)$ το πρόσημο του διαιρέτη αγνοείται αλλά το το αποτέλεσμα είναι πάντα μη αρνητικό.

4. Εκθετικές συναρτήσεις

- void *mpz_powm*)*mpz_t* rop, *mpz_t* base, *mpz_t* exp, *mpz_t* mod) Τοποθέτηση στο *rop* το $base^{exp} mod\ mod$.
- void *mpz_ui_pow_ui*(*mpz_t* rop, unsigned long int base, unsigned long int exp) Τοποθέτηση στο *rop* το $base^{exp}$. Στην περίπτωση 0^0 αποδίδει την τιμή 1.
- int $mpz\_perfect\_power\_p$(*mpz_t* op) Επιστρέφει μη μηδενική τιμή εάν ο *op* είναι τέλεια δύναμη.

5. Συναρτήσεις θεωρίας αριθμών

- void *mpz_gcd*(*mpz_t* rop, *mpz_t* op_1, *mpz_t* op_2) Τοποθέτηση στο *rop* το μέγιστο κοινο διαιρέτη του op_1 και του op_2. Το αποτέλεσμα είναι πάντα θετικό ακόμα και αν ένας ή και οι δύο παράγοντες εισόδου είναι αρνητικοί.
- void *mpz_nextprime*(*mpz_t* rop, *mpz_t* op) Τοποθέτηση στο *rop* τον επόμενο πρώτο αριθμό ο οποίος είναι μεγαλύτερος του *op*.

6. Συναρτήσεις σύγκρισεις

- int *mpz_cmp_ui*(*mpz_t* op_1, unsigned long int op_2)
  Σύγκριση των απολύτων τιμών του op_1 και του op_2. Επιστέφει θετικό αποτέλεσμα εάν $op_1 > op_2$, μηδέν εάν $op_1 = op_2$ ή αρνητικό εάν $op_1 < op_2$

7. Συναρτήσεις τυχαίων αριθμών

- void *mpz_urandomm*(*mpz_t* rop, *gmp_randstate_tstate*, *mpz_t* n)
  Εύρεση τυχαίων αριθμών οι οποίοι κυμαίνονται απο 1 μέχρι $n-1$. Η μεταβλητή *state* πρέπει να εισαχθεί με την κλήση μιας εκ των *gmp_randinit* συναρτήσεων πριν τη χρήση αυτής της συνάρτησης. Μετά το τέλος της χρήσης αυτής της μεταβλήτης χρησιμοποιείται η συνάρτηση $gmp\_randclear(state)$.
- void gmp_randseed_ui (gmp_randstate_t state, unsigned long int seed)
  Εισαγωγή τιμής *seed* στο *state*. Το μέγεθος των *seed* διευκρινίζει πόσες διαφορετικές σειρές τυχαίων αριθμών είναι πιθανών να δημιουργηθούν.

8. Διάφορες συναρτήσεις

   - int $mpz\_odd\_p(mpz\_t$ op$)$
   - int $mpz\_even\_p(mpz\_t$ op$)$
     Διευκρινίζουν εάν ο op είναι άρτιος ή περιττός. Επιστρέφουν μη μηδενικό αποτέλεσμα εάν η απάντηση είναι ναι και μηδενικό εάν είναι όχι.

5.2.2 Συναρτήσεις Αριθμών Κινητής Υποδιαστολής (χρησιμοποιούνται στον αλγόριθμο του *Miller*)

1. Συναρτήσεις Αρχικοποίησης

   - void $mpf\_init(mpf\_t$ x$)$ Εισαγωγή στο x το 0. Κάθε μεταβλητή πρέπει να εισάγεται μόνο μία φορά ή τουλάχιστον να "καθαρίζεται" κάθε φορά χρησιμοποιώντας την συνάρτηση mpf_clear μεταξύ των εισαγωγών.

2. Συναρτήσεις Ανάθεσης

   - void $mpf\_set\_z(mpf\_t$ rop, $mpf\_t\ op_1$, $mpf\_t\ op_2)$
   - void $mpf\_set\_d(mpf\_t$ rop, double op$)$ Τοποθέτηση του op στο rop

3. Αριθμητικές Συναρτήσεις

   - void $mpf\_mul(mpf\_t$ rop, $mpf\_t\ op_1$, $mpf\_t\ op_2)$
     Τοποθέτηση στο rop το γινόμενο $op_1 \times op_2$.

4. Διάφορες συναρτήσεις

   - void mpf_ceil $(mpf\_t$ rop, mpf_t op$)$: αναθέτει στη μεταβλητή rop το πάνω ακέραιο μέρος της τιμής της μεταβλητής op, δηλαδή $rop \leftarrow \lceil op \rceil$.

Βιβλιογραφία

[1] Algorithmic Number Theory, Eric Bach and Jeffrey Shallit, Volume I MIT Press.

[2] Guide to Elliptic Curve Cryptography, Darrel Hankerson, Alfred Menezes and Scott Vanstone, Springer.

[3] Probabilistic Algorithm for Testing Primality, Michael O.Rabin, Journal of number theory $12, 128 - 138(1980)$.

[4] Riemann's Hypothesis and Tests for Primality, Gary L.Miller, Journal of computer and system sciences $13, 300 - 317(1976)$

[5] Gary L.Miller, Department of Mathematics University of California, Berkeley California

[6] Pseudopowers and primality proving, Pedro Berrizbeitia, Siguna Muller and Hugh C.Williams, Volume 2 number 2 pages $219 - 236$

[7] Recognizing Primes, R.A.Moliin, International journal of Algebra vol1, 2007, no$10, 469 - 476$

[8] Σύγχρονη Κρυπτογραφία, Π.Νάστου, Π.Σπυράκης, Γ.Σταματίου, ελληνικά γράμματα

[9] Τεχνικές κρυπτογραφίας και κρυπτογράφησης, Β.Α.Κάτος, Γ.Χ.Στεφανίδης, Εκδόσεις ΖΥΓΟΣ

[10] Αλγόριθμοι, Π.Μποζάνης, Εκδόσεις Τζιόλα

[11] Στοιχεία διακριτών μαθηματικών, C.L.LIU, πανεπηστημιακές εκδόσεις Κρήτης

[12] The GNU Multiple Precision Arithmetic Library, Edition 4.2.4

[13] Υπολογιστική πολυπλοκότητα, Η.Σταυρόπουλος, 2004

[14] Switching Theory, Dr.Vicky Papadopoulou, 2008

[15] Τεχνολογίες Υλοποίησης Αλγορίθμων, Χ.Ζαρολιάγκης, ενότητα 2, Πανεπιστήμιο Πατρών

[16] Very short primality proofs, Carl Pomerance, mathematics of computation

[17] Introduction to algorithms, Rivest, Leiserson, Addison Wesley

[18] Κρυπτογραφία η επιστήμη της ασφαλούς επικοινωνίας, Δημήτριος Π. Πουλάκης, Εκδόσεις ΖΗΤΗ

[19] Data Encryption Standard (DES), FIPS PUB 46-3 U.S. Department of Commerce/National Institute of Standards and Technology.

[20] Primality Rests Based on Fermat's Little Theorem, Manindra Agrawal, Department of Computer Science Indian Institute of Technology, Kanpur.