Πανεπιστήμιο Αιγαίου
Τμήμα Μαθηματικών

Πτυχιακή Εργασία

Αλγόριθμοι Αναζήτησης Συγκρούσεων και Εφαρμογές τους στην Κρυπτανάλυση

Επιβλέπων:
Παναγιώτης Νάστου

Συγγραφέας:
Βασιλική Χριστοπούλου

Τριμελής Επιτροπή
Μεταφτσής Βασίλειος, Καθηγητής, Πρόεδρος Τμήματος Μαθηματικών
Καπόρης Αλέξιος, Επίκουρος Καθηγητής
Παναγιώτης Νάστου, Επίκουρος Καθηγητής

19 Ιουνίου 2015

Αφιερωμένη, στον παππού μου.

Χριστοπούλου Βασιλική,
Σάμος 2015.

Ευχαριστώ τον επιβλέποντα καθηγητή μου, κύριο Νάστου, για την υπομονή και την υποστήριξή του, σε όλη τη διάρκεια της προετοιμασίας της παρούσας πτυχιακής. Μέσα από την επιμονή του βελτίωσα πολλούς τομείς, τους οποίους αν και στην αρχή αμφισβητούσα, στην πορεία κατάλαβα τη σημασία τους.

Ευχαριστώ επίσης, τους δικούς μου ανθρώπους για την εμπιστοσύνη που μου έχουν δείξει, καθώς και τη φίλη μου, Νίκη, που έχει σταθεί δίπλα μου όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

Κρυπτογράφηση (encryption) ονομάζεται η διαδικασία μετασχηματισμού ενός μηνύματος σε μία ακατανόητη μορφή με τη χρήση κάποιου κρυπτογραφικού αλγορίθμου ούτως ώστε να μη μπορεί να διαβαστεί από κανέναν εκτός του νόμιμου παραλήπτη. Η αντίστροφη διαδικασία όπου από το κρυπτογραφημένο κείμενο παράγεται το αρχικό μήνυμα ονομάζεται **αποκρυπτογράφηση** (decryption).

Κρυπτογραφικός αλγόριθμος (cipher) είναι η μέθοδος μετασχηματισμού δεδομένων σε μία μορφή που να μην επιτρέπει την αποκάλυψη των περιεχομένων τους από μη εξουσιοδοτημένα μέρη. Κατά κανόνα ο κρυπτογραφικός αλγόριθμος είναι μία πολύπλοκη μαθηματική συνάρτηση όπου,

- **Αρχικό κείμενο** (plaintext) είναι το μήνυμα το οποίο αποτελεί την είσοδο σε μία διεργασία κρυπτογράφησης.
- **Κλειδί** (key) είναι ένας αριθμός αρκετών *bit* που χρησιμοποιείται ως είσοδος στην συνάρτηση κρυπτογράφησης.
- **Κρυπτοκείμενο** (ciphertext) είναι το αποτέλεσμα της εφαρμογής ενός κρυπτογραφικού αλγορίθμου πάνω στο αρχικό κείμενο.
- **Κρυπτοσύστημα** (cryptosystem) είναι ένα σύνολο κρυπτογραφικών αλγορίθμων μαζί με τη διαδικασία δημιουργίας και διαχείρισης κλειδιού.

Η κρυπτολογία ασχολείται με τη μελέτη της ασφαλούς επικοινωνίας. Κύριος στόχος της είναι να παρέχει μηχανισμούς επικοινωνίας, τα λεγόμενα κρυπτοσυστήματα, μεταξύ δύο ή περισσότερων μελών χωρίς κάποιο άλλο, μη εξουσιοδοτημένο, άτομο να κατανοεί το περιεχόμενο των μηνυμάτων που ανταλλάσσονται, να υποκλέψει όπως λέγεται τις διακινούμενες πληροφορίες.

Η κρυπτολογία αποτελείται από δύο ενότητες: την κρυπτογραφία και την κρυπτανάλυση. Η **κρυπτογραφία** είναι ο κλάδος που ασχολείται με τους μαθηματικούς

μετασχηματισμούς, που είναι απαραίτητοι για την ασφαλή μεταφορά της πληροφορίας ενώ η **κρυπτανάληση** είναι ο κλάδος που ασχολείται με την ανάλυση και το ¨σπάσιμο¨ των κρυπτοσυστημάτων, ώστε να γίνουν κατανοητές οι πληροφορίες που μεταδίδονται.

Ιστορικά, η κρυπτογράφηση μηνυμάτων σήμαινε την μετατροπή της πληροφορίας από μία κατανοητή γλώσσα σε ένα γρίφο, για την κατανόηση του οποίου απαιτούνταν κάποιος κρυφός μετασχηματισμός. Το χαρακτηριστικό των παλιότερων κρυπτοσυστημάτων ήταν η επεξεργασία της γλωσσικής δομής του κειμένου- μηνύματος. Στα νεότερα κρυπτοσυστήματα γίνεται μετατροπή του κειμένου- μηνύματος σε αριθμητικό ισοδύναμο. Το βάρος από εκεί και πέρα πέφτει σε διάφορα μαθηματικά πεδία, όπως τα διακριτά μαθηματικά, η θεωρία αριθμών, η θεωρία πληροφορίας, η υπολογιστική πολυπλοκότητα, η στατιστική και συνδυαστική ανάλυση, ώστε να είναι αδύνατο, σε πραγματικό χρόνο, να ¨σπάσει¨ το κρυπτοσύστημα και να ανακαλυφθεί τελικά το αρχικό μήνυμα.

Η ιστορία της κρυπτογραφίας ξεκινά από την εποχή των αρχαίων Αιγυπτίων περίπου 4000 χρόνια πριν. Μέχρι και τις αρχές του 20ου αιώνα οι μέθοδοι κρυπτογράφησης και αποκρυπτογράφησης χρησιμοποιούσαν χαρτί, μολύβι και στην καλύτερη περίπτωση απλούς μηχανισμούς κρυπτογράφησης και αποκρυπτογράφησης (κλασσική κρυπτογραφία). Στις αρχές του 20ου αιώνα εφευρέθηκαν πολύπλοκες μηχανές, όπως η μηχανή Enigma και Purple Machine οι οποίες χρησιμοποιήθηκαν στον δεύτερο παγκόσμιο πόλεμο από τους Γερμανούς και τους Ιάπωνες αντίστοιχα. Αργότερα η εμφάνιση των ηλεκτρονικών συστημάτων και των υπολογιστών επέτρεψε την υλοποίηση εξαιρετικά πολύπλοκων κρυπτογραφικών συστημάτων.

Η εξέλιξη της κρυπτογραφίας συμβαδίζει με την εξέλιξη της κρυπτανάλυσης. Η ανακάλυψη και εφαρμογή μεθόδων ανάλυσης της συχνότητας εμφάνισης κάθε χαρακτήρα σε κρυπτοκείμενα, έφερε το ¨σπάσιμο¨ των κωδικών και κάποτε ανέτρεψε τη ροή της ιστορίας. Είναι γνωστό ότι η αποκρυπτογάφηση του ¨τηλεγραφήματος Zimmermann¨, έφερε την εμπλοκή των ΗΠΑ στον πρώτο παγκόσμιο πόλεμο και η αποκρυπτογράφηση μηνυμάτων των χιτλερικών στρατευμάτων, από τους συμμάχους, έφερε το τέλος του δευτέρου παγκοσμίου πολέμου δυο χρόνια περίπου νωρίτερα. Παρακάτω θα αναφέρουμε τις δύο βασικές κατηγορίες κρυπτογραφικών συστημάτων. [2]

1.1 Κρυπτογραφικά Συστήματα

1.1.1 Συμμετρικά κρυπτοσυστήματα

Τα συμμετρικά κρυπτοσυστήματα είναι τα συστήματα εκείνα που χρησιμοποιούν για την κρυπτογράφηση και την αποκρυπτογράφηση ένα κοινό κλειδί K γνωστό ως *μυστικό* ή *συμμετρικό κλειδί* (secret key). Το κλειδί αυτό θα πρέπει να είναι γνωστό

μόνο στα εξουσιοδοτημένα μέλη, πιο συγκεκριμένα ο αποστολέας χρησιμοποιεί το μυστικό κλειδί για να κρυπτογραφήσει το μήνυμα ενώ ο παραλήπτης χρησιμοποιεί το ίδιο κλειδί για να το αποκρυπτογραφήσει.

Η συμμετρική κρυπτογραφία χρησιμοποιείται εδώ και χιλιάδες χρόνια. Ένας από τους παλαιότερους γνωστούς κώδικες κρυπτογραφίας είναι ο αλγόριθμος του Καίσαρα, που αποτελεί έναν απλό κώδικα αντικατάστασης. Άλλοι γνωστοί και πιο σύγχρονοι αλγόριθμοι είναι οι DES, IDEA, RC5, CAST-128 και AES.

Στα πλεονεκτήματα της συμμετρικής κρυπτογραφίας συγκαταλέγονται οι υψηλές ταχύτητες κρυπτογράφησης και αποκρυπτογράφησης που μπορούν να υπερβούν τα $100Mbps$ καθώς επίσης και οι μικρές απαιτήσεις της σε μνήμη και υπολογιστική ισχύ. Έτσι καθίσταται εφικτή η εφαρμογή της σε περιβάλλοντα όπως αυτά ενός κινητού τηλεφώνου ή μιας έξυπνης κάρτας. Τέλος το μέγεθος του κρυπτοκειμένου είναι αρκετά μικρότερο από αυτό του αρχικού μηνύματος.

Η ανάγκη της ανταλλαγής του συμμετρικού κλειδιού μεταξύ αποστολέα και παραλήπτη είναι ένας από τους σημαντικότερους περιορισμούς της συμμετρικής κρυπτογραφίας, μιας και η ασφάλειά της βασίζεται μόνο στην μυστικότητα του κλειδιού. Για να το πετύχουν αυτό, στα συμμετρικά κρυπτοσυστήματα, τα μέλη ανταλλάσσουν το κλειδί, πριν από την αποστολή του μηνύματος, μέσω ενός ασφαλούς καναλιού επικοινωνίας, το οποίο καθιστά τελικά δύσκολη την επικοινωνία, την ανταλλαγή δηλαδή της πληροφορίας, μεταξύ τους. Η δυσκολία αυτή γίνεται ακόμα μεγαλύτερη όταν ο παραλήπτης και ο αποστολέας είναι άγνωστοι μεταξύ τους. Σε αυτήν την περίπτωση προκύπτει η ανάγκη πιστοποίησης της ταυτότητας και των δύο έτσι ώστε να αποφευχθεί η διαβίβαση του κλειδιού σε κάποιο τρίτο, μη εξουσιοδοτημένο, άτομο. Ένας ακόμη περιορισμός αφορά στη δυσκολία κλιμάκωσης της μεθόδου. Καθώς το πλήθος των χρηστών που θέλουν να επικοινωνήσουν μεταξύ τους μεγαλώνει, αυξάνεται και το πλήθος των κλειδιών που θα χρησιμοποιηθούν για τις επιμέρους επικοινωνίες.

Έστω ότι μία οντότητα Β θέλει να στείλει ένα μήνυμα στην οντότητα Α με τη χρήση συμμετρικού συστήματος. Η διαδικασία που θα ακολουθήσουν είναι η εξής:

1. Ο Α επιλέγει τυχαία το κλειδί Κ μέσα από τον κλειδοχώρο.
2. Ο Α αποστέλλει το Κ στον Β μέσω ενός ασφαλούς καναλιού.
3. Ο Β δημιουργεί το μήνυμα, το κρυπτογραφεί με το Κ και το στέλνει στον Α.
4. Ο Α λαμβάνει το κρυπτογραφημένο μήνυμα και το αποκρυπτογραφεί με το Κ.

1.1.2 Ασύμμετρα κρυπτοσυστήματα

Στα μέσα της δεκαετίας του '70 οι Whitfield, Diffie και Martin Hellman πρότειναν μία νέα τεχνική για τον περιορισμό των προβλημάτων της συμμετρικής κρυπτογραφίας.

Η τεχνική αυτή, γνωστή ως *κρυπτογραφία δημόσιου κλειδιού* ή *ασύμμετρη κρυπτογραφία*, βασίζεται στην ύπαρξη ενός *ζεύγους κλειδιών* (key pair), το *ιδιωτικό* (private key) και το *δημόσιο* (public key). Το δημόσιο είναι διαθέσιμο σε όλους και χρησιμοποιείται για να κρυπτογραφηθεί ένα μήνυμα, ενώ το ιδιωτικό είναι γνωστό μόνο στον κάτοχό του και χρησιμοποιείται κατά την αποκρυπτογράφηση του μηνύματος.

Η ασύμμετρη κρυπτογραφία χρησιμοποιείται για την κρυπτογράφηση και αποκρυπτογράφηση δεδομένων καθώς και για την ψηφιακή υπογραφή τους. Η ασφάλεια αυτών των αλγορίθμων βασίζεται στη μυστικότητα του ιδιωτικού μόνο κλειδιού, η παραγωγή του οποίου είναι υπολογιστικά αδύνατη από το δημόσιο κλειδί.

Το σημαντικότερο πλεονέκτημα της ασύμμετρης κρυπτογραφίας είναι ότι δεν απαιτείται ανταλλαγή μυστικού κλειδιού. Το δημόσιο κλειδί είναι ελεύθερα διαθέσιμο, καθιστώντας έτσι τη διαχείριση των κλειδιών ευκολότερη, ενώ το ιδιωτικό κλειδί είναι γνωστό μόνο στον ιδιοκτήτη του. Παρ' όλα αυτά η ασύμμετρη κρυπτογραφία έχει μεγάλες απαιτήσεις σε υπολογιστική ισχύ και είναι αρκετά αργή όταν πρόκειται για μεγάλα μηνύματα. Η διαδικασία αποστολής μηνύματος από την οντότητα Α προς την οντότητα Β με τη χρήση ενός ασύμμετρου κρυπτοσυστήματος έχει ως εξής:

1. Η γεννήτρια κλειδιών του Α παράγει ένα ζεύγος κλειδιών e_A, d_A.
2. Η γεννήτρια κλειδιών του Β παράγει ένα ζεύγος κλειδιών e_B, d_B.
3. Ο Α και ο Β ανταλλάσσουν τα δημόσια κλειδιά τους.
4. Ο Α δημιουργεί μήνυμα $M = \{m_1, m_2, ..., m_i\}$, όπου τα $m_1, m_2, ..., m_i$ ανήκουν στον χώρο μηνυμάτων.
5. Ο Α κρυπτογραφεί το M με το δημόσιο κλειδί του Β, το παραγόμενο κρυπτοκείμενο $c = \{c_1, c_2, ..., c_j\}$ αποστέλλεται στον Β.
6. Ο Β λαμβάνει το c και με το ιδιωτικό του κλειδί αποκρυπτογραφεί το κρυπτοκείμενο c στο αρχικό μήνυμα M.

Υπάρχουν αρκετά κρυπτοσυστήματα δημόσιου κλειδιού όπως τα RSA, ElGamal καθώς και αυτά των ελλειπτικών καμπυλών.

RSA

Το κρυπτοσύστημα RSA χρησιμοποιείται τόσο για την κρυπτογράφηση όσο και για ψηφιακές υπογραφές. Το όνομά του προέρχεται από τους τρεις δημιουργούς του Ron Rivest, Adi Shamir και Leonard Adleman.

Για να παραχθούν τα κλειδιά στον RSA επιλέγονται τυχαία δύο μεγάλοι πρώτοι αριθμοί, p και q, του ίδιου μήκους στη δυαδική τους αναπαράσταση, για μέγιστη

ασφάλεια και στη συνέχεια υπολογίζεται το γινόμενό τους:

$$n = p \times q \tag{1.1}$$

Υπολογίζεται η συνάρτηση Euler $\phi(n) = (p-1)(q-1)$ και στη συνέχεια επιλέγεται επίσης τυχαία το κλειδί κρυπτογράφησης, το δημόσιο δηλαδή κλειδί e, τέτοιο ώστε το e και το $\phi(n)$ να είναι σχετικά πρώτοι, δηλαδή $(e, \phi(n)) = 1$ και $1 < e < \phi(n)$. Συνεπώς το e είναι αντιστρέψιμο στοιχείο στην ομάδα $Z^*_{\phi(n)}$.

Τέλος χρησιμοποιώντας τον επεκταμένο αλγόριθμο του Ευκλείδι βρίσκουμε το κλειδί αποκρυπτογράφησης, το ιδιωτικό δηλαδή κλειδί d.

$$\begin{aligned} e \cdot d &= 1 \bmod \phi(n) \\ d &= e^{-1} \bmod \phi(n) \end{aligned} \tag{1.2}$$

Έστω ότι η οντότητα Α θέλει να στείλει ένα μήνυμα m στην οντότητα Β με τη χρήση του RSA

1. Ο Α λαμβάνει το δημόσιο κλειδί (n, e) του Β το οποίο κατασκευάστηκε με τις εξισώσεις (1.2).
2. Μετατρέπει το m σε έναν ακέραιο ή περισσότερους ακεραίους στο διάστημα $\{0, 1, ..., n-1\}$, διασπά δηλαδή το m σε αριθμητικά μέρη μικρότερα του n.
3. Υπολογίζεται η τιμή $c = m^e \bmod n$ και αποστέλλεται στον Β.
4. Ο Β χρησιμοποιώντας το ιδιωτικό του κλειδί d υπολογίζει την ποσότητα $c^d \bmod n$ που ισούται με το αρχικό μήνυμα m.

Γνωρίζουμε ότι:

$$\begin{aligned} ed &\equiv 1 \,(\mathrm{mod}\phi(n)) \Rightarrow ed = t\phi(n) + 1 \\ c &\equiv m^e \pmod n \\ m &\equiv c^d \pmod n \end{aligned}$$

Επίσης από το θεώρημα Fermat-Euler αν $(m, n) = 1$ τότε $m^{\phi(n)} \equiv 1 \pmod n$

Χάρις στα παραπάνω στοιχεία, είμαστε σε θέση να δείξουμε με μαθηματικά επιχειρήματα τη διαδικασία απόκρυπτογράφησης του αλγόριθμου RSA.

$$\begin{aligned} y &\equiv m^{ed} \pmod n \\ &\equiv m^{t\phi(n)+1} \pmod n \\ &\equiv m^{t\phi(n)} m \pmod n \\ &\equiv m \pmod n \end{aligned}$$

Ας δούμε ένα παράδειγμα εφαρμογής του αλγορίθμου RSA:

Παράδειγμα 1.1.1 *Έστω οι τυχαία επιλεγμένοι πρώτοι αριθμοί* $p = 61$ *και* $q = 53$ *και το γινόμενό τους* $n = p \times q = 61 \cdot 53 = 3233$. *Υπολογίζουμε τη συνάρτηση Euler* $\phi(n) = (p-1)(q-1) = 3120$.

Θα επιλέξουμε τώρα έναν τυχαίο αριθμό e *τέτοιον ώστε να είναι σχετικά πρώτος με το* $\phi(n)$ *και μεγαλύτερος του 1. Έστω* $e = 17$.

Τώρα πρέπει να βρούμε d *που να ικανοποιεί την εξίσωση* $de \equiv 1 \pmod{\phi(n)}$. *Εφόσον ο* e *είναι σχετικά πρώτος με το* $\phi(n)$ *θα υπάρχει ο αντίστροφός του στο* $Z^*_{\phi(n)}$. *Άρα η παραπάνω εξίσωση γίνεται* $d \equiv e^{-1} \pmod{\phi(n)}$. *Από τον αλγόριθμο της διαίρεσης προκύπτει ότι:*

$$\begin{aligned} 3120 &= 17 \cdot 183 + 9 \\ 17 &= 9 \cdot 1 + 8 \\ 9 &= 8 \cdot 1 + 1 \\ 8 &= 1 \cdot 8 + 0 \end{aligned} \tag{1.3}$$

Τώρα θα βρούμε τον αντίστροφο του $e = 17$ *:*

$$\begin{aligned} 1 &= 9 - 8 \\ &= 9 - (17 - 9) = 9 - 17 + 9 = 2 \cdot 9 - 17 \\ &= 2(3120 - 17 \cdot 183) - 17 = 6240 - 17 \cdot 366 - 17 \\ &= 2 \cdot 3120 + (-367) \cdot 17 \end{aligned} \tag{1.4}$$

Συνεπώς βρήκαμε ότι ο αντίστροφος του $e = 17$ *στο* $Z^*_{\phi(n)}$ *είναι ο αριθμός* $3120 - 367 = 2753$. *Το δημόσιο κλειδί είναι το* $(n = 3233, e = 17)$ *και το ιδιωτικό κλειδί είναι το* $d = 2753$.

Υποθέτουτας ότι το μήνυμα που θέλουμε να κρυπτογραφήσουμε είναι το $m = 123$, *το κρυπτοκείμενο είναι ίσο με* $c = 123^{17} \pmod{3233} = 855$. *Ομοίως η αποκρυπτογράφηση του κρυπτοκειμένου γίνεται υπολογίζοντας την ποσότητα* $m = 855^{2753} \pmod{3233} = 123$.

Η ασφάλεια του RSA βασίζεται στην δυσκολία παραγοντοποίησης ενός σύνθετου ακεραίου. Εαν κάποιος γνωρίζει το n είναι εύκολο να υπολογίσει το p και το q και κατ' επέκταση το $\phi(n)$ το οποίο θα του δώσει τα e και d. Για να γίνει αυτό, αρκεί να αποσυνθέσουμε το n ως γινόμενο δύο πρώτων αριθμών p και q. Αν το n είναι αρκετά μεγάλο, δεν υπάρχει κάποιο γνωστό μέσο για να βρούμε τα p και q σε ένα λογικό χρονικό διάστημα. Σήμερα, οι πρώτοι αριθμοί που χρησιμοποιούνται για την κρυπτογράφηση μηνυμάτων ξεπερνούν τα 200 ψηφία.

ElGamal

Μετά το κρυπτοσύστημα RSA ακολούθησε το κρυπτοσύστημα ElGamal, το οποίο περιγράφηκε αρχικά από τον Taher ElGamal το 1984. Η ασφάλεια του συγκεκριμένου συστήματος βασίζεται στη δυσκολία επίλυσης του προβλήματος του διακριτού λογαρίθμου και του προβλήματος Diffie-Hellman [5].

Στο πρόβλημα του διακριτού λογαρίθμου δίνονται ένας πρώτος αριθμός p, ένας γεννήτορας g του Z_p^* και ένα στοιχείο $b \in Z_p^*$ και ζητείται να βρεθεί ακέραιος x με $0 \leq x \leq p-2$, τέτοιος ώστε $g^x \equiv b \pmod{p}$.

Η χρησιμότητα του προβλήματος του διακριτού λογαρίθμου, από την άποψη της κρυπτογραφίας έγκειται στο ότι η συνάρτηση $b = g^x$ ανήκει στην κατηγορία των μονόδρομων συναρτήσεων. Δηλαδή, ενώ είναι αλγοριθμικά εύκολος ο υπολογισμός της τιμής $b = g^x$ σε σχετικά μικρό χρονικό διάστημα για κάθε x, ο υπολογισμός της τιμής της αντίστροφης συνάρτησης $x = \log_g(b)$ σε κατάλληλα επιλεγμένες ομάδες G είναι υπολογιστικά δύσκολος. Δεν υπάρχει, δηλαδή, μέχρι σήμερα αποδοτικός αλγόριθμος για τον υπολογισμό του x δοθέντων των g, b.

Στο πρόβλημα Diffie-Hellman δίνονται ένας πρώτος αριθμός p, ένας γεννήτορας g του Z_p^* και δύο στοιχεία $a, b \in Z_p^*$. Επειδή g γεννήτορας θα ισχύει $a \equiv g^x \pmod{p}$ και $b \equiv g^y \pmod{p}$ για κάποια $x, y \in Z$. Ζητείται να βρεθεί το $c \equiv g^{xy} \pmod{p}$ γνωρίζοντας τα a, b.

Έστω λοιπόν πως θέλει να επικοινωνήσει ο Α με τον Β, τότε συμφωνούν δημόσια σε έναν μεγάλο πρώτο αριθμό έστω p και σε έναν αριθμό g, ο οποίος είναι γεννήτορας του σώματος Z_p^*, δηλαδή οι δυνάμεις του g : $g^1, g^2, ..., g^{p-1}$ παράγουν το σώμα.

Στη συνέχεια ο Α θα επιλέξει έναν τυχαίο αριθμό x_1, τον οποίο θα τον κρατήσει μυστικό, θα είναι δηλαδή το ιδιωτικό του κλειδί και θα υπολογίσει την ποσότητα $y_1 \equiv g^{x_1} \pmod{p}$. Την ίδια διαδικασία ακολουθεί κι ο Β, διαλέγει ένα τυχαίο x_2, το ιδιωτικό του κλειδί και υπολογίζει το $y_2 \equiv g^{x_2} \pmod{p}$. Αφού ολοκληρωθεί αυτή η διαδικασία ανταλλάσσουν τα y_1 και y_2 χωρίς να αποκαλύψουν τα x_1, x_2.

Τώρα ο Α θα υπολογίσει το $y_2^{x_1} \pmod{p}$ και ο Β το $y_1^{x_2} \pmod{p}$, τα οποία όμως είναι ο ίδιος αριθμός αφού:

$$\begin{aligned} y_1^{x_2} &\equiv (g^{x_1})^{x_2} \pmod{p} \\ &\equiv g^{x_1 x_2} \pmod{p} \\ &\equiv (g^{x_2})^{x_1} \pmod{p} \\ &\equiv y_2^{x_1} \end{aligned}$$

Τα μόνα που μπορεί να γνωρίζει κάποιος είναι τα p, g, y_1, y_2. Αν όμως ο p είναι ένας μεγάλος πρώτος αριθμός δεν υπάρχει αποτελεσματικός τρόπος να χρησιμοποιήσει τα στοιχεία που γνωρίζει για να υπολογίσει το $y_1^{x_2}$. Γίνεται σαφές από τον παραπάνω αλγόριθμο ότι η μυστικότητα του ιδιωτικού κλειδιού βασίζεται στη δυσκολία επίλυσης

του προβλήματος του διακριτού λογαρίθμου.

Έστω ότι η οντότητα Α θέλει να κρυπτογραφήσει ένα μήνυμα m και να το στείλει στην οντότητα Β με τη χρήση του ElGamal. Θα ακολουθήσει τα εξής βήματα:

1. Λαμβάνει το δημόσιο κλειδί (p, g, y_2) του Β.
2. Μετατρέπει το μήνυμα m σε έναν ακέραιο ή σε περισσότερους ακεραίους στο διάστημα $\{0, 1, ..., p-1\}$.
3. Επιλέγει ένα τυχαίο k,$1 \leq k \leq p-2$.
4. Υπολογίζει τις τιμές $\gamma \equiv g^k \pmod{p}$, $\delta \equiv m \cdot y_2^k \pmod{p}$ και στέλνει το κρυπτοκείμενο $c = (\gamma, \delta)$ στον Β.

Έστω τώρα ότι η οντότητα Β θέλει να αποκρυπτογραφήσει το μήνυμα $c = (\gamma, \delta)$ που έλαβε από την Α, τότε:

1. Υπολογίζει την τιμή $m' \equiv \gamma^{p-1-x_2} \pmod{p}$.
2. Ανακτά το αρχικό μήνυμα m υπολογίζοντας την τιμή $m' \cdot \delta \pmod{p}$.

Ας εξετάσουμε τώρα γιατί λειτουργεί σωστά ο αλγόριθμος ElGamal. Γνωρίζουμε ότι:

$$\begin{aligned} \gamma &\equiv g^k \pmod{p} \\ \delta &\equiv m \cdot (g^{x_2})^k \pmod{p} \\ \phi(p) &\equiv p-1 \Rightarrow \\ m^{\phi(p)} &\equiv 1 \pmod{n} \end{aligned}$$

Από τα παραπάνω παίρνουμε ότι:

$$\begin{aligned} m &\equiv m' \cdot \delta \pmod{p} \\ &\equiv \gamma^{p-1-x_2} \cdot \delta \pmod{p} \\ &\equiv (\gamma^{p-1})^{-x_2} \cdot \delta \pmod{p} \\ &\equiv \gamma^{p-1} \cdot \gamma^{-x_2} \cdot \delta \pmod{p} \\ &\equiv \gamma^{\phi(p)} \cdot \gamma^{-x_2} \cdot \delta \pmod{p} \\ &\equiv 1 \cdot \gamma^{-x_2} \cdot \delta \pmod{p} \\ &\equiv \gamma^{-x_2} \cdot \delta \pmod{p} \\ &\equiv g^{-kx_2} \cdot m(g^{x_2})^k \pmod{p} \\ &\equiv g^{-kx_2+kx_2} \cdot m \cdot \pmod{p} \\ &\equiv g^0 \cdot m \pmod{p} \\ &\equiv m \cdot \pmod{p} \end{aligned}$$

Ας δούμε ένα παράδειγμα εφαρμογής του αλγορίθμου ElGamal:

Παράδειγμα 1.1.2 *Έστω ο πρώτος αριθμός* $p = 47$, *ο γεννήτορας του σώματος* Z_p^* $g = 2$ *και ο τυχαία επιλεγμένος αριθμός* $x_2 = 10$. *Βρίσκουμε το* $y_2 = g^{x_2} \pmod p = 2^{10} \pmod{47} = 1024 \pmod{47} = 37$.

Η οντότητα Α κρυπτογραφεί το m *και το στέλνει στην Β.*

1. *Λαμβάνει το δημόσιο κλειδί του Β το* $(p, g, y_2) = (47, 2, 37)$.

2. *Έστω* $m = 25$.

3. *Επιλέγει τυχαίο* k *τέτοιο ώστε* $0 \leq k \leq 47 - 1 = 46$, *έστω* $k = 36$.

4. *Υπολογίζει τις τιμές:* $\gamma \equiv g^k \pmod p \equiv 2^{36} \pmod{47} = 14$, $\delta \equiv m \cdot (g^{x_2})^k \pmod p \equiv 25(2^{10})^{36} \pmod{47} = 37$ *και στέλνει το κρυπτοκείμενο* $c = (\gamma, \delta) = (14, 37)$ *στον Β.*

Η οντότητα Β τώρα αποκρυπτογραφεί το μήνυμα $c = (14, 37)$.

1. $m' \equiv \gamma^{p-1-x_2} \pmod p \equiv 14^{46-10} \pmod{47} = 14^{36} \pmod{47} = 21$

2. $m \equiv m' \cdot \delta \pmod p \equiv 21 \cdot 37 \pmod{47} \equiv 777 \pmod{47} = 25$

Το βασικό μειονέκτημα του αλγορίθμου κρυπτογράφησης είναι η διόγκωση μηνύματος, δηλαδή το γεγονός ότι το κρυπτοκείμενο c έχει διπλάσιο μέγεθος από το αρχικό μήνυμα m.

Κατά τη διάρκεια κρυπτογράφησης είναι ιδιαίτερα σημαντικό να χρησιμοποιούνται διαφορετικοί ακέραιοι k για διαφορετικά αρχικά μηνύματα. Έστω ότι χρησιμοποιήσουμε τον ίδιο ακέραιο k για την κρυπτογράφηση δύο διαφορετικών μηνυμάτων m_1 και m_2, τότε τα αντίστοιχα κρυπτοκείμενα (γ_1, δ_1) και (γ_2, δ_2) θα έχουν την εξής ιδιότητα:

$$\frac{\delta_1}{\delta_2} = \frac{m_1}{m_2} \tag{1.5}$$

Επομένως αν κάποιο από τα m_1, m_2 είναι γνωστό, είναι πολύ εύκολο από (1.5) να βρεθεί και η τιμή του άλλου.

Ελλειπτικές καμπύλες στο $\Re$

Το 1985 μία παραλλαγή του προβλήματος του διακριτού λογαρίθμου προτάθηκε από τους Miller και Koblitz που όριζε το πρόβλημα στην ομάδα που αποτελείται από τα σημεία μίας ελλειπτικής καμπύλης. Το πρόβλημα αυτό ονομάστηκε *πρόβλημα διακριτού λογαρίθμου σε ελλειπτικές καμπύλες.*

Ορισμός 1.1.1 *Μία ελλειπτική καμπύλη EC πάνω σε ένα σώμα F περιγράφεται από την εξίσωση του Weierstrass:*

$$y^2 + a_1xy + a_2y = x^3 + a_3x^2 + a_4x + a_5$$

όπου $a_1, ..., a_5 \in F$ *και* $x, y \in F$.

Μη ιδιάζουσα ελλειπτική καμπύλη EC ονομάζεται η καμπύλη με εξίσωση:

$$y^2 = x^3 + ax + b$$

όπου $a, b \in \Re$ *τέτοια ώστε:* $4a^3 + 27b^2 \neq 0$. *Η προηγούμενη σχέση διασφαλίζει την ύπαρξη τριών διακριτών ριζών της εξίσωσης :* $x^3 + ax + b = 0$.

- Μία ελλειπτική καμπύλη είναι συμμετρική ως προς τον άξονα x, οπότε μπορούμε να ορίσουμε το αντίθετο σημείο $-P$, ενός σημείου $P = (x_1, y_1)$, να είναι το $(x_1, -y_1)$.

- Το σημείο στο άπειρο O_∞ είναι το σημείο στο οποίο συναντιόνται οι παράλληλες προς τον άξονα y ευθείες. Επειδή τα P, $-P$ και O_∞ ανήκουν στην ίδια ευθεία, ορίζεται εύκολα ότι: $P + (-P) = O_\infty$ και $P + O_\infty = P$. Δηλαδή το σημείο στο άπειρο είναι το ουδέτερο στοιχείο της πρόσθεσης σημείων στην EC.

- Το σύνολο EC με την πράξη της πρόσθεσης όπως ορίστηκε παραπάνω είναι μία αβελλιανή ομάδα.

Θεωρούμε τώρα δύο διαφορετικά σημεία, έστω P=(x_1, y_1) και Q=(x_2, y_2) της EC κι έστω η ευθεία $y = \lambda x + c$ με κλίση $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$, η οποία τέμνει την EC στα σημεία αυτά. Αντικαθιστώντας την εξίσωση της ευθείας στην EC παίρνουμε :

$$(\lambda x + c)^2 = x^3 + ax + b \tag{1.6}$$

Η (1.6) είναι τριτοβάθμια εξίσωση με δύο από τις ρίζες της τα x_1 και x_2. Υπάρχει όμως και μία τρίτη ρίζα που αντιστοιχεί στο σημείο $-R$= $(x_3, -y_3)$=$(x_3, -(\lambda x_3 + c))$. Ορίζουμε $P + Q = R$. Συνεπώς η ευθεία τέμνει την EC σε τρία σημεία. Για να βρούμε τις συντεταγμένες του σημείου R ακολουθούμε την εξής διαδικασία:

Αντικαθιστώ στην EC το y με $y = \lambda x + c$ και έχω

$$x^3 - \lambda^2 x^2 + (a - 2\lambda c)x + b - c^2 = 0$$

δεδομένου ότι τα x_1, x_2 είναι ρίζες της EC έχουμε:

$$x^3 = \lambda^2 - x_2 - x_1$$

και άρα

$$y_3 = \lambda(x_1 - x_3) - y_1$$

Οι παραπάνω σχέσεις προέκυψαν για διαφορετικά σημεία P και Q.

Στην περίπτωση όπου $Q = -P = (x_1, -y_1)$, η κλίση γίνεται άπειρη, γεγονός που μας οδηγεί στο σημείο O_∞.

Τέλος στην περίπτωση όπου $P = Q$, η ευθεία που περνάει από τα σημεία είναι η εφαπτόμενη στο P (ή Q). Τα x_3, y_3 δίνονται από τις ίδιες σχέσεις που περιγράφηκαν παραπάνω. Το μόνο που αλλάζει είναι το λ, η κλίση της ευθείας. Η κλίση μίας ευθείας σε ένα σημείο της ελλειπτικής καμπύλης είναι η τιμή της παραγώγου της καμπύλης σε αυτό το σημείο. Παραγωγίζοντας ως προς x και τα δύο μέλη της EC προκύπτει ότι:

$$\begin{aligned} 2yy' &= 3x^2 + a \Rightarrow \\ y' &= \frac{3x^2 + a}{2y} \end{aligned}$$

Η κλίση επομένως της εφαπτόμενης ευθείας στο σημείο $P = (x_1, y_1)$ είναι ίση με:

$$\lambda = \frac{3x_1^2 + a}{2y_1}.$$

Ελλειπτικές καμπύλες στο Z_p

Οι ελλειπτικές καμπύλες οι οποίες έχουν κρυπτογραφικό ενδιαφέρον είναι ορισμένες στο σώμα Z_p, όπου p πρώτος και $p > 3$. Η πράξη της πρόσθεσης ορίζεται με τον ίδιο τρόπο όπως στο $\mathfrak{R}$, με τη διαφορά ότι οι πράξεις είναι *modulo* και θα ασχοληθούμε με μη ιδιάζουσες ελλειπτικές καμπύλες. Συνεπώς, η ελλειπτική καμπύλη ορισμένη στο Z_p, για κάποιον πρώτο ακέραιο $p > 3$, είναι το σύνολο των λύσεων $(x, y) \in Z_p \times Z_p$ της ισοτιμίας $y^2 \equiv x^3 + ax + b \pmod p$, όπου $a, b \in Z_p$ και $4a^3 + 27b^2 \not\equiv 0 \pmod p$ μαζί με το σημείο στο άπειρο O_∞.

Μία άλλη σημαντική ιδιότητα των ελλειπτικών καμπυλών στο Z_p, είναι ότι τα σημεία της EC μαζί με το O_∞ ορίζουν κυκλική υποομάδα. Οποιοδήποτε δηλαδή, σημείο ανήκει στην EC, εκτός του O_∞, είναι γεννήτορας αυτής. Δηλαδή, δοθέντος

κάποιου σημείου P της καμπύλης, η διαδοχική πρόσθεση του P στον εαυτό του, θα διατρέξει όλα τα σημεία της EC. Έτσι αν $P \in$ EC τότε μπορούμε να γράψουμε ότι

$$\text{EC} =< P >= \{Q = kP : 0 \leq k \leq \mid \text{EC} \mid -1\}$$

Στο πρόβλημα του διακριτού λογαρίθμου στις ελλειπτικές καμπύλες έχουμε μία ελλειπτική καμπύλη EC(Z_p), ένα σημείο $P \in$ EC τάξεως n και ένα σημείο $Q \in < P >$ και ψάχνουμε να βρούμε ένα $0 \leq k \leq n-1$ τέτοιο ώστε $Q = kP$.

Κάθε χρήστης Α ακολουθεί τα παρακάτω βήματα για τη δημιουργία του δημόσιου και ιδιωτικού κλειδιού.

1. Αρχικά, όλοι οι χρήστες έχουν συμφωνήσει στη χρήση του ίδιου σώματος Z_p και της ίδιας ελλειπτικής καμπύλης EC(Z_p).
2. Κάθε χρήστης υπολογίζει ένα τυχαίο σημείο P στην EC με μεγάλη τάξη.
3. Επιλέγει έναν τυχαίο ακέραιο k μικρότερο από την τάξη του σημείου P και υπολογίζει το $Q = kP$.
4. Το δημόσιο κλειδί του Α είναι το (Q, P) και το ιδιωτικό είναι ο ακέραιος k.

Έστω ότι η οντότητα Α θέλει να κρυπτογραφήσει ένα μήνυμα m, με τη βοήθεια του κρυπτοσυστήματος ελλειπτικών καμπυλών και να το στείλει στην οντότητα Β, θα ακολουθήσει τα εξής βήματα:

1. Επιλέγει έναν αριθμό $0 < k < n-1$, όπου n είναι ο μεγαλύτερος πρώτος παράγοντας της τάξης m της EC.
2. Υπολογίζει το σημείο $R = kP$, όπου P είναι το σημείο βάσης με το οποίο κατασκευάστηκε το δημόσιο κλειδί του χρήστη Β.
3. Υπολογίζει το σημείο $Q = kQ_B$, όπου $Q_B = k_B P$, όπου Q_B είναι το δημόσιο κλειδί του Β και k_B είναι το ιδιωτικό του κλειδί.
4. Υπολογίζει τον ακέραιο $c = zx \pmod{p}$, όπου z είναι η αναπαράσταση του μηνύματος m που κρυπτογραφείται σε μορφή ακέραιου αριθμού, x είναι η x συντεταγμένη του σημείου Q και p είναι η τάξη του πεπερασμένου σώματος στο οποίο ορίζεται η EC.
5. Το ζευγάρι (R, c) αποτελεί την κρυπτογραφημένη μορφή του m.

Έστω τώρα ότι η οντότητα Β θέλει να αποκρυπτογραφήσει το (R, c), που έλαβε από την οντότητα Α, θα ακολουθήσει τα παρακάτω βήματα:

1. Υπολογίζει το σημείο $Q = k_B R$.

2. Υπολογίζει τον ακέραιο $z = c \cdot x^{-1} \pmod{p}$, όπου x είναι η τετμημένη του Q.

3. Ο ακέραιος z είναι η αναπαράσταση του m σε μορφή ακεραίου και ακολουθώντας την αντίστροφη διαδικασία με την οποία τα μηνύματα μετατρέπονται σε ακεραίους, προκύπτει το m.

Κεφάλαιο 2

Αλγόριθμοι και Πολυπλοκότητα

Πρόβλημα ονομάζεται ένα ερώτημα που πρέπει να απαντηθεί [3]. Περιγράφεται από παραμέτρους ή ελεύθερες μεταβλητές. Θέτοντας συγκεκριμένες τιμές στις παραμέτρους ενός προβλήματος προκύπτει ένα στιγμιότυπο του προβλήματος. **Αλγόριθμος** ονομάζεται μία καλώς ορισμένη, υπολογιστική διαδικασία, η οποία καλείται να επιλύσει κάθε στιγμιότυπο ενός προβλήματος, εντός πεπερασμένου χρόνου.

Ας εξετάσουμε το πρόβλημα του περιοδεύοντος πωλητή. Έστω ότι ένας έμπορος θέλει να επισκεφθεί m πόλεις οι οποίες συνδέονται μεταξύ τους με συγκεκριμένο τρόπο. Ο έμπορος ξεκινώντας από μια πόλη αναζητεί εκείνη τη διαδρομή που θα διασχίσει όλες τις πόλεις ακριβώς μια φορά και θα επιστρέψει στην πόλη από την οποία ξεκίνησε. Επιπρόσθετα, το μήκος της διαδρομής θέλει να είναι το ελάχιστο δυνατό.

Οι παράμετροι του προβλήματος είναι το σύνολο των πόλεων $C = \{c_1, c_2, ..., c_m\}$ καθώς και οι αποστάσεις $d(c_i, c_j)$ για κάθε ζεύγος πόλεων c_i και c_j. Αναζητούμε έναν αλγόριθμο που να ελαχιστοποιεί την ποσότητα :

$$\left(\sum_{i=1}^{m-1} d(c_{p(i)}, c_{p(i+1)}) \right) + d(c_{p(m)}, c_{p(1)})$$

Ο παραπάνω τύπος εκφράζει τη διαδρομή που αρχίζει από την πόλη $c_{p(1)}$, περνάει από όλες τις πόλεις ακριβώς μία φορά και επιστρέφει από την τελευταία πόλη $c_{p(m)}$ στην αρχική.

Για να μπορέσουμε να εκφράσουμε το πρόβλημα σε έναν υπολογιστή πρέπει να κωδικοποιήσουμε τα στιγμιότυπά του, αυτό το κατορθώνουμε χρησιμοποιώντας μια γλώσσα L, ένα σύνολο δηλαδή συμβολοσειρών, με σύμβολα από ένα πεπερασμένο σύνολο συμβόλων, το αλφάβητο Σ.

2.1 Υπολογιστικά Μοντέλα

Στη θεωρία υπολογισμού μελετάμε το κατά πόσο είναι εφικτό να λυθεί ένα πρόβλημα με χρήση κάποιου αλγορίθμου σε ένα υπολογιστικό μοντέλο. Υπολογιστικό μοντέλο είναι ένας εξιδανικευμένος υπολογιστής. Ανάλογα με το πρόβλημα που θέλουμε να επιλύσουμε χρησιμοποιούμε και το αντίστοιχο υπολογιστικό μοντέλο. Τα πιο συνηθισμένα υπολογιστικά μοντέλα είναι το πεπερασμένο αυτόματο καθώς και η μηχανή Turing. [4]

2.1.1 Ντετερμινιστικό Πεπερασμένο Αυτόματο (DFA)

Ντετερμινιστικό Πεπερασμένο Αυτόματο (deterministic finite automata), ονομάζεται μία πεντάδα, $(Q, \Sigma, \delta, q_0, F)$, όπου:

1. Q είναι ένα πεπερασμένο σύνολο, όπου τα στοιχεία του ονομάζονται καταστάσεις.
2. Σ είναι ένα πεπερασμένο σύνολο, το οποίο ονομάζεται αλφάβητο.
3. $\delta : Q \times \Sigma \to Q$ είναι μία συνάρτηση μεταβάσεων από τη μία κατάσταση στην άλλη.
4. q_0 είναι η κατάσταση έναρξης.
5. $F \subseteq Q$ είναι το σύνολο καταστάσεων αποδοχής.

Κάθε αυτόματο όταν λάβει μία λέξη εισόδου, επεξεργάζεται τα σύμβολά της ένα προς ένα, από τα αριστερά προς τα δεξιά και παράγει μία έξοδο. Η επεξεργασία ξεκινάει από την κατάσταση έναρξης. Μετά την ανάγνωση κάθε συμβόλου, το αυτόματο μεταβαίνει από την τρέχουσα κατάσταση σε κάποια άλλη, βάσει της δ. Αφού ολοκληρώσει την ανάγνωση όλων των συμβόλων της λέξης παράγει την έξοδό του, εάν βρίσκεται σε κατάσταση αποδοχής η έξοδός του είναι αποδοχή, διαφορετικά απόρριψη.

Εάν A είναι το σύνολο των λέξεων που αποδέχεται ένα αυτόματο M, λέμε ότι το A είναι η γλώσσα του M ή το M αναγνωρίζει την A και γράφουμε $L(M) = A$.

Μία γλώσσα ονομάζεται **κανονική**, αν υπάρχει πεπερασμένο αυτόματο που να την αναγνωρίζει.

2.1.2 Μη Ντετερμινιστικό Πεπερασμένο Αυτόματο (NFA)

Μη Ντετερμινιστικό Πεπερασμένο Αυτόματο (non deterministic finite automata), ονομάζεται μία πεντάδα, $(Q, \Sigma, \delta, q_0, F)$, όπου:

1. Q είναι ένα πεπερασμένο σύνολο καταστάσεων.
2. Σ είναι ένα πεπερασμένο αλφάβητο.
3. $\delta : Q \times \Sigma_\varepsilon \to P(Q)$ είναι η συνάρτηση μεταβάσεων.
4. $q_0 \in Q$ είναι η κατάσταση έναρξης.
5. $F \subseteq Q$ είναι το σύνολο καταστάσεων αποδοχής.

Οι NFA μηχανές λειτουργούν όπως οι DFA με κάποιες διαφορές. Όταν η DFA βρίσκεται σε κάποια δεδομένη κατάσταση και διαβάζει το επόμενο σύμβολο η επόμενη κατάστασή της είναι μονοσήμαντα καθορισμένη, σε αντίθεση με την NFA, στην οποία σε κάθε κατάσταση της ενδέχεται να υπάρχουν περισσότερες από μία επιλογές για την επόμενη κατάσταση. Στο NFA συναντάμε και το σύμβολο ε, το οποίο δεν ανήκει στο αλφάβητο και αναπαριστά την κενή λέξη.

Έστω λοιπόν ότι φτάνουμε σε μία κατάσταση από την οποία έχουμε για δεδομένο σύμβολο περισσότερες από μία δυνατότητες για τη συνέχεια. Αφού η NFA διαβάσει το σύμβολο, διασπάται σε τόσα αντίγραφα του εαυτού της, όσες και οι δυνατότητες συνέχειας και τις ακολουθεί όλες παράλληλα. Κάθε αντίγραφο ακολουθεί μία από τις δυνατές διαδρομές και συνεχίζει όπως η αρχική μηχανή. Εάν σε κάποιο αντίγραφο το επόμενο σύμβολο δεν εμφανίζεται τότε παύει να είναι ενεργό. Εάν πάλι βρεθεί σε κατάσταση η οποία μεταβαίνει σε κάποια άλλη μέσω του ε, χωρίς να διαβάσει κανένα σύμβολο της λέξης εισόδου, διασπάται σε δύο αντίγραφα. Το ένα από αυτά παραμένει στην τρέχουσα κατάσταση και το άλλο ακολουθεί το ε. Η NFA αποδέχεται την είσοδό της, εάν στο τέλος της ανάγνωσης υπάρχει κάποιο αντίγραφο, το οποίο βρίσκεται σε κατάσταση αποδοχής.

Για κάθε μη ντετερμινιστικό πεπερασμένο αυτόματο, υπάρχει ισοδύναμο ντετερμινιστικό. Μία γλώσσα λέγεται κανονική αν και μόνο αν υπάρχει μη ντετερμινιστικό πεπερασμένο αυτόματο που να την αναγνωρίζει.

2.2 Μηχανές Turing

Θα εξετάσουμε τώρα ένα πιο ισχυρό μοντέλο που προτάθηκε από τον Alan Turing το 1936 και λέγεται μηχανή Turing.

2.2.1 Ντετερμινιστική Μηχανή Turing (DTM)

Η ντετερμινιστική μηχανή Turing μοιάζει πολύ με το πεπερασμένο αυτόματο, αλλά διαθέτει άπειρη μνήμη, για την οποία χρησιμοποιεί μία άπειρη ταινία και επιπλέον έχει δύο αλφάβητα, το αλφάβητο εισόδου, Σ και το αλφάβητο της ταινίας, Γ, στο

οποίο συμπεριλαμβάνεται το σύμβολο του διαστήματος, ⊔. Διαθέτει μία κεφαλή, η οποία έχει τη δυνατότητα, μετακινούμενη επάνω στην ταινία, να διαβάζει και να γράφει σύμβολα. Στην αρχή η ταινία περιέχει μόνο τη λέξη εισόδου και σύμβολα διαστήματος σε όλες τις υπόλοιπες θέσεις και η κεφαλή της είναι τοποθετημένη στο αριστερότερο κελί της ταινίας. Η κατάσταση αυτή είναι η κατάσταση έναρξης, q_0. Σε κάθε υπολογιστικό βήμα, η μηχανή διαβάζει το σύμβολο του τρέχοντος κελιού και βάσει της τρέχουσας κατάστασης q, υπολογίζεται η συνάρτηση δ. Για να αποθηκεύσει κάποιες πληροφορίες, η μηχανή της γράφει σε κάποιο τμήμα της ταινίας και για να της διαβάσει μετακινείται στο αντίστοιχο τμήμα της. Η μηχανή συνεχίζει τον υπολογισμό μέχρι να φτάσει σε κάποια κατάσταση αποδοχής ή απόρριψης και να παράξει την αντίστοιχη έξοδο. Εάν δεν μεταβεί ποτέ σε κάποια από αυτές τις καταστάσεις, συνεχίζει τη λειτουργία της χωρις να τερματίζει ποτέ.

Τυπικότερα μία ντετερμινιστική μηχανή Turing ορίζεται ως μία εφτάδα, (Q, Σ, Γ, δ, q_0, q_{accept},q_{reject}), όπου

1. Q είναι το πεπερασμένο σύνολο καταστάσεων.
2. Σ είναι το πεπερασμένο αλφάβητο εισόδου.
3. Γ είναι το πεπερασμένο αλφάβητο ταινίας, με ⊔ ∈ Γ και Σ ⊆ Γ.
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{A,\Delta\}$ είναι η συνάρτηση μεταβάσεων.
5. $q_0 \in Q$ είναι η κατάσταση έναρξης.
6. $q_{accept} \in Q$ είναι η κατάσταση αποδοχής.
7. $q_{reject} \in Q$ είναι η κατάσταση απόρριψης και ισχύει $q_{accept} \neq q_{reject}$.

2.2.2 Πολυταινιακή Μηχανή Turing

Μία πολυταινιακή μηχανή Turing είναι μία ντετερμινιστική μηχανή Turing, με περισσότερες από μία ταινίες, όπου κάθε ταινία έχει τη δική της κεφαλή εγγραφής και ανάγνωσης. Η μόνη διαφορά είναι στη συνάρτηση μεταβάσεων, η οποία έχει την εξής μορφή:

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{A, \Delta, \Sigma\}^k,$$

όπου k το πλήθος των ταινιών. Η παραπάνω έκφραση μας δηλώνει ότι η εγγραφή, η ανάγνωση καθώς και η μετακίνηση της κεφαλής γίνεται ταυτόχρονα σε όλες τις ταινίες. Η έκφραση:

$$\delta(q_i, \alpha_1, ..., \alpha_k) = (q_j, b_1, ..., b_k, A, \Delta, ..., A),$$

σημαίνει ότι αν η μηχανή βρίσκεται στην κατάσταση q_i και οι κεφαλές 1 έως k διαβάζουν τα α_1 έως α_k αντίστοιχα, τότε η μηχανή μεταβαίνει στην κατάσταση q_j, γράφει στις ταινίες τα σύμβολα b_1 έως b_k και ανάλογα με το αντίστοιχο σύμβολο μετακινεί την κεφαλή ή αριστερά (Α) ή δεξιά (Δ) ή την αφήνει ακίνητη (Σ).

Για κάθε πολυταινιακή μηχανή Turing υπάρχει ισοδύναμη μονοταινιακή μηχανή Turing, δηλαδή αναγνωρίζουν την ίδια γλώσσα. Από το παραπάνω προκύπτει ότι μία γλώσσα είναι αναγνωρίσιμη αν και μόνο αν υπάρχει πολυταινιακή μηχανή Turing που να την αναγνωρίζει.

2.2.3 Μη ντετερμινιστική μηχανή Turing (NTM)

Η διαφορά της NTM από την κλασσική μηχανή Turing, είναι ότι ενδέχεται να έχει περισσότερες από μία δυνατότητες για τη συνέχεια, δηλαδή η συνάρτηση μεταβάσεών της δίνεται από τον τύπο ([4]):

$$\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{A, \Delta\}).$$

Ο υπολογισμός αυτής της μηχανής γίνεται διασπώντας τη σε αντίγραφα όσες και οι διαφορετικές δυνατότητες. Η μηχανή αποδέχεται την είσοδό της εάν υπάρχει κάποιο αντίγραφο, το οποίο οδηγεί σε κατάσταση αποδοχής, διαφορετικά την απορρίπτει. Για κάθε NTM, υπάρχει ισοδύναμη DTM. Μία γλώσσα λέγεται αναγνωρίσιμη εάν υπάρχει μηχανή Turing που να την αναγνωρίζει.

Μία μηχανή Turing μπορεί να αποδεχτεί, να απορρίψει ή να εγκλωβιστεί (loop). Με τον όρο εγκλωβισμός, εννοούμε ότι η μηχανή δεν τερματίζει ποτέ. Μία μηχανή, η οποία δεν εγκλωβίζεται ποτέ, ονομάζεται διαγνώστης και λέμε ότι διαγιγνώσκει την γλώσσα που αναγνωρίζει.

Όταν ζητάμε να λύσουμε ένα πρόβλημα στην ουσία αναζητούμε ένα διαγνώστη αυτού του προβλήματος. Εάν δεν μπορούμε να βρούμε αλγόριθμο που να μην εγκλωβίζεται η μηχανή, τότε το πρόβλημα μας είναι άλυτο. Εάν η μηχανή τερματίζει είτε με αποδοχή είτε με απόρριψη, τότε το πρόβλημα μας επιδέχεται λύση.

2.3 Χρονική Πολυπλοκότητα

Η χρονική πολυπλοκότητα εξαρτάται από το πλήθος των βημάτων που εκτελεί ο αλγόριθμος. Ο χρόνος εκτέλεσης του αλγορίθμου υπολογίζεται συναρτήσει του μήκους της λέξης εισόδου το οποίο απεικονίζει το μέγεθος του προβλήματος. Ακολουθούμε την ανάλυση χειρότερης περίπτωσης, στην οποία λαμβάνουμε υπόψιν μόνο το μέγιστο από τους χρόνους εκτέλεσης για όλες τις εισόδους συγκεκριμένου μήκους.

Έστω M μία αιτιοκρατική μηχανή Turing που τερματίζει για κάθε είσοδο. Ο **χρόνος εκτέλεσης** ή η **χρονική πολυπλοκότητα** της M είναι η συνάρτηση f :

$N \to N$ που για κάθε n επιστρέφει ως $f(n)$ το μέγιστο πλήθος βημάτων που είναι δυνατόν να πραγματοποιήσει η Μ όταν το μήκος της εισόδου της είναι n.

Συχνά για εισόδους μεγάλου μήκους είναι περίπλοκο να ορίσουμε τον ακριβή χρόνο εκτέλεσης του αλγορίθμου. Για το λόγο αυτό χρησιμοποιούμε την ασυμπτωτική ανάλυση, με την οποία βρίσκουμε μία εκτίμηση του χρόνου αυτού. Λαμβάνουμε υπόψιν μόνο τους όρους που μεγαλώνουν γρήγορα όσο το n μεγαλώνει, δηλαδή μας ενδιαφέρει μόνο ο μεγιστοβάθμιος όρος, χωρίς τον συντελεστή του. Για παράδειγμα στην συνάρτηση $f(n) = 6n^3 + 2n^2 + 20n + 45$ μας ενδιαφέρει μόνο το n^3 και λέμε ότι η f είναι ασυμπτωτικά το πολύ n^3. Συμβολικά γράφουμε $f(n) = O(n^3)$ και ονομάζεται ασυμπτωτικός συμβολισμός ή συμβολισμός του κεφαλαίου όμικρον.

Έστω δύο συναρτήσεις $f, g : N \to R^+$. Λέμε ότι $f(n) = O(g(n))$, εάν υπάρχουν θετικοί ακέραιοι c και n_0, τέτοιοι ώστε για κάθε ακέραιο $n \geq n_0$, να ισχύει

$$f(n) \leq c \cdot g(n).$$

Όταν $f(n) = O(g(n))$, λέμε ότι η $g(n)$ αποτελεί ασυμπτωτικό άνω φράγμα της $f(n)$.

Σχετικός με τον παραπάνω συμβολισμό είναι και ο συμβολισμός του πεζού όμικρον. Έστω f και g δύο συναρτήσεις από το σύνολο N στο σύνολο R^+. Λέμε ότι $f(n) = o(g(n))$, εάν ισχύει ότι:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Με άλλα λόγια, η έκφραση $f(n) = o(g(n))$, σημαίνει ότι για οποιονδήποτε πραγματικό αριθμό $c > 0$, υπάρχει ακέραιος n_0 τέτοιος ώστε $f(n) < c \cdot g(n)$ για κάθε $n \geq n_0$.

Έστω $t : N \to R^+$ μία συνάρτηση. Ορίζουμε ως κλάση χρονικής πολυπλοκότητας $TIME(t(n))$ τη συλλογή όλων των γλωσσών που μπορούν να διαγνωστούν από κάποια μηχανή Turing σε χρόνο το πολύ $t(n)$.

2.4 Κλάσεις πολυπλοκότητας

Η θεωρία πολυπλοκότητας ταξινομεί τα προβλήματα σε κλάσεις πολυπλοκότητας, ανάλογα με τη δυσκολία τους. Εύκολο ονομάζεται το πρόβλημα, το οποίο μπορεί να λυθεί εντός πολυωνυμικού χρόνου, ενώ δύσκολο ονομάζεται το πρόβλημα του οποίου η λύση απαιτεί μη πολυωνυμικό χρόνο (τα προβλήματα αυτά αναφέρονται ως εκθετικού χρόνου). Ο διαχωρισμός αυτός οφείλεται στο γεγονός ότι οι αλγόριθμοι πολυωνυμικού χρόνου είναι αρκετά γρήγοροι ώστε να είναι χρήσιμοι, σε αντίθεση με τους αλγορίθμους εκθετικού χρόνου κι αυτό γιατί ο ρυθμός αύξησης ενός πολυωνύμου είναι πολύ μικρότερος από αυτόν μίας μη πολυωνυμικής συνάρτησης. Ιδιαίτερου ενδιαφέροντος στο πλαίσιο αυτό είναι οι κλάσεις P (Deterministic

Polynomial Time) και NP (Non Deterministic Polynomial Time). Οι κλάσεις αυτές ορίζονται ως προβλήματα απόφασης, δηλαδή προβλήματα στα οποία καλούμαστε να απαντήσουμε μια συγκεκριμένη ερώτηση με ναι ή όχι.

2.4.1 Η κλάση P

Η κλάση P αποτελείται από τις γλώσσες που μπορούν να διαγνωστούν σε πολυωνυμικό χρόνο από κάποια αιτιοκρατική μηχανή Turing. Συμβολικά:

$$P = \bigcup_k TIME(n^k)$$

Για να δείξουμε ότι ένα πρόβλημα ανήκει στην κλάση P πρέπει να κάνουμε δύο πράγματα. Πρώτον, θα πρέπει να προσδιορίσουμε ένα πολυωνυμικό, ασυμπτωτικό άνω φράγμα της χρονικής συνάρτησης του αλγορίθμου. Δεύτερον, θα πρέπει να εξετάσουμε μεμονωμένα το κάθε του στάδιο για να διαπιστώσουμε αν καθένα από αυτά υλοποιείται σε πολυωνυμικό χρόνο.

2.4.2 Η κλάση NP

Ονομάζουμε επαληθευτή μίας γλώσσας A, οποιονδήποτε αλγόριθμο V, τέτοιον ώστε: A= { w| ο V αποδέχεται τη λέξη $\langle w, c\rangle$, όπου c κάποια λέξη }. Ο χρόνος εκτέλεσης του επαληθευτή υπολογίζεται συναρτήσει του μήκους της λέξης w. Επομένως, ένας επαληθευτής πολυωνυμικού χρόνου έχει πολυωνυμικό χρόνο εκτέλεσης ως προς το μήκος της w. Τότε η γλώσσα A θα ονομάζεται πολυωνυμικά επαληθεύσιμη. Ο επαληθευτής V προκειμένου να επαληθεύσει τη συμμετοχή του w στην A χρησιμοποιεί μια λέξη c η οποία ονομάζεται **πιστοποιητικό**.

NP είναι η κλάση όλων των γλωσσών που επιδέχονται επαληθευτή πολυωνυμικού χρόνου. Ένα πρόβλημα απόφασης λέμε ότι επιλύεται σε μη-ντετερμινιστικό πολυωνυμικό χρόνο αν υπάρχει ένας μη-ντετερμινιστικός αλγόριθμος που, εκμεταλλευόμενος μια τυχαία επιλογή, μπορεί σε πολυωνυμικό χρόνο να δώσει καταφατική απάντηση στο πρόβλημα. Η κλάση αυτών των προβλημάτων συμβολίζεται με NP (Nondeterministic Polynomial time).

Δεδομένης μιας συνάρτησης $t : N \to N$, το σύνολο των γλωσσών που αποφασίζονται σε μη ντετερμινιστικό χρόνο $O(t(n))$ αποτελούν την $NTIME(t(n))$ κλάση μη ντετερμινιστικής χρονικής πολυπλοκότητας που ορίζεται ως: $NTIME(t(n))$= { L| η γλώσσα L που διαγιγνώσκεται από κάποια μη ντετερμινιστική μηχανή Turing χρόνου $O(t(n))$ }.

2.4.3 Η κλάση NP-complete

Οι Stephen Cook και Leonid Levin ανακάλυψαν για ορισμένα προβλήματα στην κλάση NP, ότι η πολυπλοκότητα καθενός από αυτά συνδέεται με την πολυπλοκότητα ολόκληρης της κλάσης. Συγκεκριμένα, εάν καταφέρουμε να δείξουμε ότι κάποιο από αυτά τα προβλήματα επιδέχεται αλγόριθμο πολυωνυμικού χρόνου, τότε θα ισχύει το ίδιο για όποιο άλλο πρόβλημα της NP. Τα προβλήματα αυτά ονομάζονται NP-complete, ή NP-πλήρη.

Ανακεφαλαιώνοντας τα προβλήματα της κλάσης P επιλύονται σε πολυωνυμικό χρόνο, απαντώντας με ένα ναι ή όχι, τα προβλήματα της κλάσης NP επαληθεύουν το ναι σε πολυωνυμικό χρόνο ενώ στην κλάση NP-complete ανήκουν τα δύσκολα προβλήματα της κλάσης NP, για τα οποία δεν έχει βρεθεί ακόμα κάποιος αποδοτικός πολυωνυμικός αλγόριθμος.

2.5 Πιθανοτικοί Αλγόριθμοι

Οι πιθανοτικοί αλγόριθμοι παίζουν σημαντικό ρόλο στην επιστήμη της πληρφορικής και αναπτύσσονται καθημερινά. Ένας πιθανοτικός αλγόριθμος αποτελείται από βήματα στα οποία γίνονται τυχαίες επιλογές και βασίζονται σε τυχαίες γεννήτριες αριθμών. Συγκεκριμένα ένας πιθανοτικός αλγόριθμος μπορεί να τερματίσει σε δύο διαφορετικά αποτελέσματα για το ίδιο στιγμιότυπο. Υπάρχουν πολλά σημαντικά προβλήματα των οποίων η λύση βασίζεται σε αυτούς τους αλγόριθμους, όπως το πρόβλημα εύρεσης δύο πρώτων παραγόντων ενός σύνθετου αριθμού.

Οι πιθανοτικοί αλγόιθμοι χωρίζονται σε τέσσερεις κατηγορίες :

1. Τυχαιοποιήσεις των ντετερμινιστικών αλγορίθμων
2. Monte - Carlo αλγόριθμοι
3. Las - Vegas αλγόριθμοι
4. Αριθμητικοί πιθανοτικοί αλγόριθμοι

Σε μία τυχαιοποίηση ενός ντετερμινιστικού αλγορίθμου αντικαθιστούμε τα βήματα στα οποία γίνονται κανονικές επιλογές, με βήματα στα οποία γίνονται τυχαίες επιλογές. Η χειρότερη απόδοση ενός αλγορίθμου, (worst - case), δεν εμφανίζεται για κάθε στιγμιότυπό του, (input), αλλά για μερικά από αυτά. Συνεπώς, διαπιστώνουμε ότι υπάρχει μία σχέση ανάμεσα στη χειρότερη απόδοση ενός αλγορίθμου και σε ένα στιγμιότυπό του, όμως δε γνωρίζουμε τον τρόπο που επηρεάζονται. Για το λόγο αυτό εισάγουμε την τυχαιότητα σε ένα ντετερμινιστικό αλγόριθμο, περιμένοντας να σπάσει αυτή η σχέση καθώς και να ομοιογενοποιηθεί η αναμενόμενη συμπεριφορά του, (expected behavior), για κάθε στιγμιότυπο.

Οι Monte - Carlo αλγόριθμοι βρίσκουν πάντα λύση, όποιο και αν είναι το στιγμιότυπο, αλλά με πιθανοτική προσέγγιση, δεν είναι δηλαδή βέβαιο ότι θα είναι σωστή. Αντιθέτως οι Las - Vegas αλγόριθμοι δεν δίνουν λύση πάντα, αν όμως δώσουν λύση αυτή είναι πάντα σωστή. Τέλος, οι αριθμητικοί πιθανοτικοί αλγόριθμοι ήταν από τα πρώτα παραδείγματα εισαγωγής της τυχαιοποίησης στο σχεδιασμό των αλγόριθμων. Ένα κλασσικό παράδειγμα είναι η εκτίμηση του αριθμού π.

Στους πιθανοτικούς αλγορίθμους κάνουμε μία τυχαία επιλογή από ένα σύνολο στοιχείων. Για παράδειγμα για δοσμένους ακέραιους m, n με $m < n$, υποθέτουμε ότι υπάρχει μία τυχαία διαδικασία $(m, ..., n, j)$, η οποία αναθέτει στο j έναν τυχαίο ακέραιο μεταξύ των m και n και με κάθε ακέραιο να είναι εξίσου πιθανό να επιλεγεί. Επιπλέον οι διαδοχικές επιλογές είναι ανεξάρτητες μεταξύ τους. Για ένα δοσμένο διάστημα πραγματικών αριθμών (a, b), υποθέτουμε πάλι ότι υπάρχει μία διαδικασία, $((a, b), x)$, η οποία αναθέτει στο x έναν τυχαία επιλεγμένο πραγματικό αριθμό, μεταξύ των a και b. Η κατανομή του x είναι ομοιόμορφη και ανεξάρτητη του διαστήματος (a, b).

Επειδή οι γεννήτριες τυχαίων αριθμών δεν είναι πάντα διαθέσιμες βασιζόμαστε σε ψευδοτυχαίες γεννήτριες. Μία γεννήτρια ψευδοτυχαίων αριθμών είναι μία ντετερμινιστική διαδικασία, βάσει της οποίας από ένα δοσμένο αριθμό, ο οποίος ονομάζεται **σπόρος**, παράγει μία ακολουθία αριθμών η οποία μοιάζει με την ακολουθία που θα προέκυπτε έπειτα από διαδοχικές αναθέσεις τιμών σε μία γνήσια τυχαία γεννήτρια αριθμών. Ωστόσο για δύο διαφορετικές αρχικές τιμές, μία ψευδοτυχαία γεννήτρια αριθμών παράγει δύο πανομοιότυπες ακολουθίες.

Η απόδοση ενός πιθανοτικού αλγορίθμου δεν εξαρτάται μόνο από την είσοδό του, αλλά και από τα αποτελέσματα των τυχαίων επιλογών του. Όπως αναφέραμε ένας πιθανοτικός αλγόριθμος μπορεί να τερματίσει σε δύο διαφορετικά αποτελέσματα για το ίδιο στιγμιότυπο, έστω I, οπότε θα έχουμε και διαφορετικούς αριθμούς βασικών λειτουργιών. Στην περίπτωση αυτή ο αριθμός των βασικών λειτουργιών που απαιτούνται για την επίλυση του I, $\tau(I)$, δεν είναι πλέον καλά καθορισμένος, επομένως στηριζόμαστε στον αναμενόμενο αριθμό βασικών λειτουργιών, $\tau_{exp}(I)$. Έπειτα από πολλές επαναλήψεις του αλγορίθμου με ένα σταθερό στιγμιότυπο, I, αναμένουμε να εκτελεστεί ο αλγόριθμός μας με $\tau_{exp}(I)$ βασικές λειτουργίες κατά μέσο όρο. Εάν ο αλγόριθμος εκτελεί πολλές τυχαίες επιλογές, τότε ακόμα και για μία επανάληψη αναμένουμε πως ο βασικός αριθμός λειτουργιών προσεγγίζεται από τον $\tau_{exp}(I)$. Με τον ίδιο τρόπο προσεγγίζονται και η βέλτιστη απόδοση, (best - case), B_n, η χειρότερη απόδοση, (worst - case), W_n, καθώς και η μέση πολυπλοκότητα, (average complexities), A_n, από $B_{exp}(n), W_{exp}(n)$ και $A_{exp}(n)$ αντίστοιχα, τα οποία ορίζονται παρακάτω.

$$\begin{aligned} B_{exp}(n) &= \min\{\tau_{exp}(I), I \in I_n\} \\ W_{exp}(n) &= \max\{\tau_{exp}(I), I \in I_n\} \\ A_{exp}(n) &= E\{\tau_{exp}\}, \end{aligned}$$

όπου I_n το σύνολο όλων των στιγμιοτύπων του αλγορίθμου με μέγεθος n.

2.5.1 Τυχαιοποιήσεις Ντετερμινιστικών Αλγορίθμων

Οι αλγόριθμοι αυτοί προκύπτουν από ντετερμινιστικούς αλγορίθμους, όπου σε μερικά βήματα των οποίων εισάγουμε τυχαιότητα. Επιπλέον, οι τυχαιοποιήσεις των ντετερμινιστικών αλγορίθμων τείνουν να προκαλέσουν την προσέγγιση της αναμενόμενης συμπεριφοράς κάθε στιγμιοτύπου, στη μέση συμπεριφορά, δηλαδή $A_{exp}(n) \rightarrow A(n)$. Συμπεραίνουμε επομένως ότι σε μία τυχαιοποίηση έχουμε καλύτερο αποτέλεσμα εάν ισχύει :

$$A(n) < W(n)$$

Έτσι μία κατάλληλη τυχαιοποίηση συνήθως οδηγεί σε $W_{exp}(n)$, το οποίο είναι μικρότερο από $W(n)$.

Ένα παράδειγμα αυτής της κατηγορίας αλγορίθμων είναι ο αλγόριθμος ταξινόμησης **Quick - Sort**, ο οποίος βασίζεται στην τεχνική «διαίρει και βασίλευε». Ο αλγόριθμος αυτός για μία ακολουθία n στοιχείων, έχει μέση πολυπλοκότητα ίση με $\Theta(n \cdot \log n)$ αλλά η χειρότερη απόδοσή του ισούται με $\Theta(n^2)$, δηλαδή κατά μέσο όρο είναι αρκετά γρήγορος. Μία τυχαιοποίηση του Quick - Sort δεν εξαλείφει την πιθανότητα ότι για ένα στιγμιότυπο μπορούν να εκτελεστούν $\Omega(n^2)$ βασικές λειτουργίες, αλλά σπάει τη σύνδεση μεταξύ του στιγμιοτύπου και της χειρότερης απόδοσης του αλγορίθμου. Συγκεκριμένα για μία τυχαιοποίηση του Quick - Sort και για κάθε στιγμιότυπο μεγέθους n θα ισχύει :

$$\tau_{exp}(I) = A(n),$$

όπου $A(n)$ η μέση συμπεριφορά του μη - τυχαιοποιημένου Quick - Sort.

Τέλος, η τυχαιοποίηση του Quick - Sort οδηγεί στην παρακάτω ομογενοποίηση:

$$\boxed{B_{exp}(n) = A_{exp}(n) = W_{exp}(n)} \tag{2.1}$$

Οι τυχαιοποιήσεις αλγορίθμων που επιτυγχάνουν την παραπάνω ομογενοποίηση ονομάζονται Sherwood, από τον Robin Hood.

2.5.2 Monte - Carlo Αλγόριθμοι

Ένας Monte - Carlo αλγόριθμος είναι ένας πιθανοτικός αλγόριθμος, του οποίου η πιθανότητα να τερματίσει σε σωστό αποτέλεσμα, ανεξαρτήτως του στιγμιοτύπου, είναι συγκεκριμένη και αυξάνεται με τις επαναλαμβανόμενες δοκιμές. Συνήθως προτιμάμε η πιθανότητα αυτή να είναι μεγαλύτερη από κάποια θετική σταθερά για κάθε στιγμιότυπο. Συγκεκριμένα για $p \in \Re$, με $0 < p < 1$, ένας p - correct Monte - Carlo αλγόριθμος είναι ένας πιθανοτικός αλγόριθμος που δίνει σωστό αποτέλεσμα με πιθανότητα $> p$. Δυστυχώς δεν υπάρχει κάποια αποδοτική μέθοδος για να ελέγξουμε αν το αποτέλεσμα, για συγκεκριμένο στιγμιότυπο, είναι σωστό.

Ένας Monte - Carlo αλγόριθμος για ένα πρόβλημα απόφασης είναι false - biased, εάν επιστρέφοντας την τιμή false δίνει πάντα σωστό αποτέλεσμα, ενώ όταν επιστρέφει την τιμή true έχει μικρή πιθανότητα λάθους. Ανίστοιχα ορίζεται ο true - biased Monte - Carlo αλγόριθμος.

Κεφάλαιο 3

Αλγόριθμοι Αναζήτησης Συγκρούσεων

Έστω μία συνάρτηση $f : S \to S$, με S πεπερασμένο σύνολο και $x_0 \in S$. Παράγουμε την ακολουθία :

$$x_0, x_1 = f(x_0), x_2 = f(x_1), ..., x_{l+c-1} = f(x_{l+c-2}).$$

Η παραπάνω ακολουθία μπορεί να γραφεί διαφορετικά ως εξής:

$$f^0(x), f^1(x), f^2(x), ..., f^{l+c-1}(x),$$

με :

$$\begin{aligned} f^0(x) &= x = x_0 \\ f^1(x) &= f(x) = f(x_0) = x_1 \\ f^2(x) &= f(f(x)) = f(x_1) = x_2 \\ f^{l+c-1}(x) &= f(f(f(...(x)))) = f(x_{l+c-2}) = x_{l+c-1}. \end{aligned}$$

Αφού το S είναι πεπερασμένο τότε η ακολουθία είναι κυκλική, δηλαδή υπάρχουν l, c τέτοια ώστε να ισχύει $f^{l+c}(x) = f^l(x)$. Συνεπώς, για κάθε $i \geq l$ έχουμε $f^{i+c}(x) = f^i(x)$.

Ορίζουμε ως **πρόβλημα ανίχνευσης κύκλου ή σύγκρουσης**, το πρόβλημα εύρεσης των τιμών l και c, για δοσμένη συνάρτηση f και δοσμένη αρχική τιμή $x_0 \in S$. **Μήκος κύκλου** της ακολουθίας ονομάζεται ο αριθμός c, ενώ **μήκος της ουράς** ή **μήκος του οδηγού** της ακολουθίας ονομάζεται ο αριθμός l.

Τα στοιχεία $f^0(x), f^1(x), ..., f^{l-1}(x)$ αποτελούν τον **οδηγό της συνάρτησης** f **στο** x ενώ ο **κύκλος της συνάρτησης** f στο x αποτελείται από τα $f^l(x), f^{l+1}(x), ..., f^{l+c-1}(x)$. Για λόγους ευκολίας τον αριθμό $l + c$ θα τον συμβολίζουμε με n.

Το πρόβλημα του κύκλου προέκυψε στην προσπάθειά να αναλυθεί η αποτελεσματικότητα κάποιων γεννητριών τυχαίων αριθμών. Οι γεννήτριες αυτές παράγουν διαδοχικές, τυχαίες τιμές, εφαρμόζοντας μία συνάρτηση στη προηγούμενη τιμή της ακολουθίας. Λύνοντας το πρόβλημα του κύκλου βρίσκουμε τον αριθμό των διακριτά τυχαίων αριθμών που παράγονται από μια δοσμένη αρχική τιμή. Μπορούμε να δημιουργήσουμε συναρτήσεις με μηδενικούς οδηγούς και μέγιστο κύκλο. Οι συναρτήσεις όμως αυτές συνήθως έχουν δύσκολη εφαρμογή. Για να μπορέσουμε να εξετάσουμε και να ελέγξουμε τα χαρακτηριστικά μιας γεννήτριας τυχαίων αριθμών χρειαζόμαστε έναν αλγόριθμο που να επιλύει το πρόβλημα του κύκλου. Στην κρυπτογραφία, η επίλυση του προβλήματος ανίχνευσης κύκλου ή συκρούσεων χρησιμοποιείται σε μεθόδους επίλυσης θεμελιωδών προβλημάτων στα οποία βασίζεται η ασφάλεια γνωστών ασύμμετρων αλγορίθμων κρυπτογράφησης καθώς και στη σχεδίαση μονόδρομων συναρτήσεων κατακερματισμού (hash functions).

Μία μέθοδος για την ανίχνευση του κύκλου δόθηκε από τον Floyd. Η ιδέα αυτού του αλγορίθμου βασίζεται στη χρήση δύο μεταβλητών, οι οποίες παίρνουν διαδοχικές τιμές και η μία αυξάνεται με τη διπλάσια ταχύτητα από την άλλη.

Στη συνέχεια του κεφαλαίου υπολογίζεται ένα κάτω φράγμα της χρονικής πολυπλοκότητας των αλγορίθμων που επιλύουν το πρόβλημα εύρεσης σύγκρουσης ή κύκλου και παρουσιάζονται ο αλγόριθμος του Floyd [1], ο αλγόριθμος του Brent [1], ο οποίος είναι μια βελτιωμένη έκδοση του αλγόριθμου του Floyd και ο αλγόριθμος του Sedgewick [8].

3.1 Κάτω φράγμα για το πρόβλημα εύρεσης σύγκρουσης ή κύκλου

Σε αυτό το εδάφιο υπολογίζεται ένα κάτω φράγμα της χρονικής πολυπλοκότητας των αλγορίθμων που επιλύουν το πρόβλημα εύρεσης σύγκρουσης ή κύκλου. Μελετώντας τη χρονική πολυπλοκότητα μπορούμε να μάθουμε πόσο καλύτερα μπορεί να τρέξει ο αλγόριθμός μας. Θα δείξουμε ότι για οποιοδήποτε αλγόριθμο που θα λύσει το πρόβλημα του κύκλου ανεξαρτήτως της συνάρτησης f, το κάτω φράγμα είναι το ίδιο.

Θεώρημα 3.1.1 *Έστω* ***A*** *ένας αλγόριθμος για το πρόβλημα εύρεσης του κύκλου και* (f, x) *ένα στιγμιότυπο του προβλήματος με λύση το ζεύγος* (l, c)*. Τότε ο* ***A*** *με είσοδο το ζευγάρι* (f, x)*, υπολογίζει την* $f(x)$ *τουλάχιστον* $l + c$ *φορές.*

Απόδειξη 3.1.1.1 *Έστω* $y_1, y_2, ..., y_i$ *οι τιμές του πεδίου ορισμού, στις οποίες ο* ***A*** *υπολογίζει την* $f(x)$ *για το στιγμιότυπο* (f, x)*. Ο αλγόριθμος υπολογίζει τη συνάρτηση τουλάχιστον μία φορά και για αυτόν το λόγο μπορούμε να θεωρήσουμε* $i > 0$ *και* $l + c > 1$*, για το υπόλοιπο της απόδειξης.*

*Ο μόνος τρόπος για να λάβει ο **A** πληροφορίες για την $f(x)$ είναι δειγματολειπτώντας τις τιμές της συνάρτησης f, για διάφορες τιμές του x. Συνεπώς κάθε συνάρτηση η οποία συμφωνεί με την f στα $y_1, y_2, ..., y_i$, (δηλαδή διαφορετικές συναρτήσεις έχουν τις ίδιες τιμές), πρέπει να έχει την ίδια περίοδο και τον ίδιο οδηγό με την f.*

Εάν $i < l + c$ είναι δυνατόν να βρούμε μία αντίφαση κατασκευάζοντας μία συνάρτηση f', η οποία συμφωνεί στα y που είναι κάτω από $l + c$, με την f, έχει όμως διαφορετικό οδηγό. Για να κατασκευάσουμε μία τέτοια συνάρτηση, θεωρούμε τα στοιχεία $f^0(x), f^1(x), ..., f^{l+c-1}(x)$. Η παραπάνω ακολουθία έχει $l+c$ διακεκριμένα στοιχεία. Εάν $i < l + c$ από την αρχή του περιστερώνα θα υπάρχει κάποιο i_0, για το οποίο να ισχύει $0 < i_0 \leq l + c - 1$, όπου το $f^{i_0}(x)$ δεν εμφανίζεται στην ακολουθία $y_1, y_2, ..., y_i$.

Εάν $l = 0$, τότε η f' προσδιορίζεται από τα $f'(f^{i_0}(x)) = f(x)$ και $f'(z) = f(z)$ για $z \neq f^{i_0}(x)$. Η συνάρτηση αυτή έχει οδηγό 1. Οπότε καταλήξαμε σε άτοπο αφού είχαμε υποθέσει ότι $l = 0$.

Εάν $l > 0$, τότε η f' προσδιορίζεται από τα $f'(f^{i_0}(x)) = x$ και $f'(z) = f(z)$ για $z \neq f^{i_0}(x)$. Η συνάρτηση αυτή όμως έχει ηγέτη 0. Επομένως καταλήγουμε πάλι σε άτοπο αφου στην υπόθεση έχουμε l θετικό.

*Συνεπώς ο αλγόριθμος **A** δεν μπορεί να λύσει το πρόβλημα του κύκλου για $i < l+c$, άρα θα πρέπει να πάρουμε $i \geq l + c$.*

3.2 Ο Αλγόριθμος Εύρεσης Συγκρούσεων του Floyd

Έστω $S = \{0, 1, 2, ..., N - 1\}$, όπου N ένας μεγάλος θετικός ακέραιος και $f : S \to S$ μία υπολογιστικά εύκολη συνάρτηση. Για τον υπολογισμό ψευδοτυχαίων αριθμών χρησιμοποιούμε τον τύπο:

$$x_{i+1} = f(x_i) \tag{3.1}$$

Δεδομένου ότι το σύνολο S είναι πεπερασμένο, θα υπάρχουν $m \geq 0$ και $n \geq 1$ τέτοια ώστε:

$$x_{m+n} = x_m \tag{3.2}$$

Από τις (3.1) και (3.2) έχουμε:

$$\begin{aligned} x_{m+n+1} &= f(x_{m+n}) \\ &= f(x_m) \\ &= x_{m+1} \end{aligned} \tag{3.3}$$

Συνεπώς, $\forall i \geq m$ ισχύει η:

$$x_{i+n} = x_i \tag{3.4}$$

Η ελάχιστη τιμή του n ονομάζεται **περίοδος**, ενώ η ελάχιστη τιμή του m είναι το μήκος του μη περιοδικού τμήματος της ακολουθίας x_i. Η ιδέα του αλγορίθμου του Floyd για τον υπολογισμό ενός πολλαπλάσιου του n είναι να βρεθεί $j \leq m+n$, τέτοιο ώστε :

$$x_{2j} = x_j \tag{3.5}$$

Στο Σχήμα 3.1 παρουσιάζεται ο αλγόριθμος εντοπισμού ενός j, για το οποίο η (3.5) είναι αληθής.

Είσοδος: Η συνάρτηση $f : S \to S$ και ένα $x_0 \in S$.
Έξοδος: Ένα j τέτοιο ώστε : $x_{2j} = x_j$ καθώς και την περίοδο n.

1. $x := x_0$
2. $y := x_0$
3. $j := 0$
4. **Επανέλαβε:**
5. $\quad j := j + 1, x := f(x)$
6. $\quad y := f(f(y))$
7. **Μέχρις ότου:** $x = y$
8. $y := x$
9. $i := 0$
10. **Επανέλαβε:**
11. $\quad i := i + 1$
12. $\quad y := f(y)$
13. **Μέχρις ότου:** $x = y$
14. Τότε $n := i$

Σχήμα 3.1: Ο αλγόριθμος εύρεσης συγκρούσεων του Floyd.

Τα βήματα $1-3$ ονομάζονται βήματα αρχικοποίησης, καθώς δίνουν αρχικές τιμές στις μεταβλητές. Δεδομένου ότι η υπολογιστική πολυπλοκότητα καθορίζεται από τον αριθμό των εκτελούμενων υπολογισμών σε κάθε βήμα του αλγορίθμου διακρίνουμε τρεις υπολογισμούς. Οι υπολογισμοί αυτοί είναι οι $f(x), f(y), f(f(y))$. Το βήμα 5 μας λέει ότι το j αυξάνεται κάθε φορά κατά μία μονάδα, το x παίρνει την τιμή της εικόνας της προηγούμενης τιμής του και το y γίνεται η εικόνα της εικόνας της προηγούμενης τιμής του. Ο βρόχος θα επαναλαμβάνεται μέχρι $x = y$ (βήμα 7). Στα βήματα $9-12$ παρουσιάζεται ο τρόπος με τον οποίο βρίσκουμε την περίοδο n της ακολουθίας x_j, με τη βοήθεια του δείκτη σύγκρουσης j, τον οποίο έχουμε βρει στον προηγούμενο βρόχο. Αρχίζουμε δηλαδή, από το x_j και κάθε φορά βρίσκουμε

τον επόμενο στην ακολουθία και να αυξάνουμε το δείκτη i μέχρις ότου να βρούμε $x_j = x_{j+i}$. Η ποσότητα αυτή είναι η ζητούμενη περίοδος. Τέλος ο αρχικός βρόχος με τον οποίο εντοπίζεται ο δείκτης σύγκρουσης θα εκτελεστεί j φορές και θα έχει $W_F = 3j$ βήματα υπολογισμού (σε κάθε βήμα τρεις υπολογισμούς της συνάρτησης f της οποίας το χρόνο υπολογισμού δεν το λαμβάνουμε υπόψη).

Ας κοιτάξουμε πιο αναλυτικά τον παραπάνω αλγόριθμο.

1. $x = x_0, y = x_0, j = 0$
2. $j = 0 + 1 = 1, x = f(x_0) = x_1, y = f(f(x_0)) = f(x_1) = x_2$
3. $j = 1 + 1 = 2, x = f(x_1) = x_2, y = f(f(x_2)) = f(x_3) = x_4$

Παρατηρούμε ότι για $x = x_2$ έχω $y = x_{2\cdot 2}$ και εξετάζεται η ισχύς της (3.5). Επιπλέον αν στην (3.4) βάλουμε $i = j$ έχουμε $x_{j+n} = x_j$ και αν $j = n$ παίρνουμε $x_{2n} = x_n$. Άρα το j είναι πολλαπλάσιο του n, $j = k \cdot n$, $k \in \mathbf{Z}$.

3.2.1 Ανάλυση πολυπλοκότητας του αλγορίθμου

Θα περιγράψουμε τώρα την χειρότερη περίπτωση, worst-case, για τον αλγόριθμο του Floyd. Ο αλγόριθμος του Floyd τερματίζει σε :

$$j = \begin{cases} m & m \equiv 0 \pmod{n}, m > 0 \\ m + n - (m \mod n) & otherwise \end{cases}$$

Από την έκφραση $m \equiv 0 \pmod{n}$ εξάγουμε το συμπέρασμα ότι αφού το υπόλοιπο της διαίρεσης του m με το n είναι 0, το m θα είναι πολλαπλάσιο του n. Επίσης όπου $m \mod n = m - n\lfloor \frac{m}{n} \rfloor$, το υπόλοιπο δηλαδή της διαίρεσης του m με το n για $m > n$. Αφαιρώντας το υπόλοιπο αυτό από το άθροισμα $m + n$ καταφέραμε το j να είναι πολλαπλάσιο του n.

Παράδειγμα 3.2.1 *Ας παρουσιάσουμε μια εφαρμογή του παραπάνω τύπου. Έστω $m = 7$ και $n = 5$. Το υπόλοιπο της διαίρεσης $\frac{m}{n}$ είναι διάφορο του μηδενός, οπότε έχουμε*

$$j = 7 + 5 - 2 = 10$$

Από την (3.4) έχουμε $x_{i+n} = x_i$, $\forall i \geq 7$

- $i = 7$, $x_{12} = x_7$
- $i = 8$, $x_{13} = x_8$

- $i = 9$, $x_{14} = x_9$
- $i = 10$, $x_{15} = x_{10}$
- $i = 11$, $x_{16} = x_{11}$
- $i = 12$, $x_{17} = x_{12}$
- $i = 13$, $x_{18} = x_{13} = x_8$, *όμως* $18 \neq 2 \cdot 8$
- $i = 14$, $x_{19} = x_{14} = x_9$, *όμως* $19 \neq 2 \cdot 9$
- $i = 15$, $x_{20} = x_{15} = x_{10}$ *και* $20 = 2 \cdot 10$

Οπότε η (3.5) ισχύει για $j = 10$ ■

Συνεχίζοντας την προηγούμενη ανάλυση, ο αριθμός των εκτελούμενων υπολογισμών είναι $W_F = 3 \cdot j$, οπότε έχουμε

$$3\max(m,n) \leq W_F \leq 3(m+n) \tag{3.6}$$

Συνεπώς, η υπολογιστική πολυπλοκότητα του αλγόριθμου είναι $O(\max(m,n))$.

3.3 Ο Αλγόριθμος Εύρεσης Συγκρούσεων του Brent

Στο εδάφιο αυτό παρουσιάζουμε τον αλγόριθμο εύρεσης σύγκρουσης που προτάθηκε από τον Brent [1] ο οποίος είναι ταχύτερος από τον αλγόριθμο του Floyd. Αν $\varrho > 1$ μια ελεύθερη παράμετρος τότε ο αλγόριθμος εύρεσης συγκρουσης B_ϱ δίνεται στο Σχήμα 3.2.

Τα βήματα $1-4$ είναι τα βήματα αρχικοποίησης. Το βήμα 6 μας λέει ότι το x παίρνει κάθε φορά την τελευταία τιμή του y και το j την τελευταία τιμή του k, καθώς αυξάνουμε κατά μία δύναμη το προηγούμενο r (βήμα 7). Στον εσωτερικό βρόχο (βήμα 9), το k αυξάνεται κατά 1 και στο y ανατίθεται η τιμή της εικόνας της προηγούμενης τιμής του, μέχρις ότου το x να γίνει y ή το k να ξεπεράσει ή να είναι ίσο με το r (βήματα $10-11$). Ο λόγος που φράσσουμε το k είναι γιατί εάν $k \geq r$ τότε ο εσωτερικός βρόχος θα συνεχιστεί επ' άπειρον. Ο αλγόριθμός μας κάποια στιγμή θα τερματίσει δίνοντας μας την περίοδο n της ακολουθίας x_j η οποία δίνεται από τον τύπο $k-j$, δηλαδή η διαφορά της τρέχουσας τιμής του k όταν βρέθηκε η σύγκρουση με την προηγούμενη τιμή του όταν το $k \geq r$ και η οποία έχει αποθηκευτεί στη j.

Η επιλογή του u στην πραγματικότητα δεν είναι ουσιώδης. Έστω ότι ο εξωτερικός βρόχος εκτελείται s φορές, στην έκφραση της περιόδου $n = k - j$, για να τερματίσει ο αλγόριθμος πρέπει $k < r = \varrho^{u+s}$. Διαλέγουμε το k να είναι ο αμέσως μικρότερος

Είσοδος: Η συνάρτηση $f : S \to S$, το πρώτο στοιχείο της ακολουθίας $x_0 \in S$, καθώς και $\varrho > 1$ μία ελεύθερη παράμετρος και $u \in [0, 1)$.
Έξοδος: Η περίοδος n της ακολουθίας x_j.

1. $y := x_0$
2. $r := \varrho^u$
3. $k := 0$
4. $done := false$
5. **Επανέλαβε:**
6. $\quad x := y, j := k$
7. $\quad r := \varrho \times r$
8. $\quad$ **Επανέλαβε:**
9. $\quad\quad k := k + 1, y := f(y)$
10. $\quad\quad done := (x = y)$
11. $\quad$ **Μέχρις ότου:** $done$ **or** $(k \geq r)$
12. **Μέχρις ότου:** $done$
13. $n := k - j$.

Σχήμα 3.2: Ο αλγόριθμος εύρεσης συγρούσεων του Brent.

ακέραιος από το r, συνεπώς δεν μας ενδιαφέρει αν το r είναι ρητός ή άρρητος. Επομένως ακόμα και αν το u πάρει κάποια κλασματική τιμή δεν μας δημιουργεί πρόβλημα. Για το λόγο αυτό συνήθως μπορούμε να πάρουμε $u = 0$, όπου μας διευκολύνει. Επιπλέον στο ϱ δίνουμε την τιμή 2, ούτως ώστε να έχουμε δυνάμεις του 2, μιας και το ψηφιακό σύστημα έχει δύο στοιχεία.

Παράδειγμα 3.3.1 *Ας δούμε ένα παράδειγμα του αλγόριθμου του Brent. Για $y = x_0$, $r = \varrho^u$ και $k = 0$ παίρνουμε $x = x_0$, $j = 0$ και $r = \varrho^{u+1}$. Στον εσωτερικό βρόχο έχουμε $k = 1$, $y = f(x_0) = x_1$. Ελέγχουμε εάν $x_0 = x_1$. Αν ισχύει ο αλγόριθμος μας τερματίζει με $n = 1 - 0 = 1$. Αν δεν ισχύει η παραπάνω ισότητα συνεχίζουμε να δίνουμε τιμές στο k μέχρις ότου να γίνει $x = y$ ή $k \geq r$. Έστω ότι συνεχίζοντας τις δοκιμές φτάνουμε στο $k = 3 = r$ και δεν έχουμε το επιθυμητό αποτέλεσμα, $x_0 = y = x_3$, τότε επανερχόμαστε στον αρχικό μας βρόχο και βάζουμε $x = x_3$, $j = 3$ και $r = \varrho^{u+2}$. Συνεχίζουμε στον εσωτερικό βρόχο με $k = 4$, $y = f(x_3) = x_4$ και επαναλαμβάνουμε την ίδια διαδικασία όπως προηγουμένως. Έστω ότι $x_3 = x_4$, τότε η περίοδός μας είναι $n = 4 - 3 = 1$.*

3.3.1 Ανάλυση πολυπλοκότητας του αλγόριθμου του Brent

Στο εδάφιο αυτό αναλύεται η πολυπλοκότητα χειρότερης περίπτωσης του αλγορίθμου B_ϱ του Brent. Ας υποθέσουμε ότι ο εξωτερικός βρόχος, του αλγορίθμου B_ϱ, εκτελείται

$s \geq 1$ φορές. Τότε ο αλγόριθμος τερματίζει σε :

$$j = \begin{Bmatrix} 0, & s = 1 \\ \lceil \varrho^{u+s-1} \rceil, & s > 1 \end{Bmatrix} \tag{3.7}$$

Όπου $j = \lceil \varrho^{u+s-1} \rceil = \frac{r}{\varrho}$, μιας και $\varrho^{u+s-1} = \varrho^{u+s} \cdot \varrho^{-1} = \varrho^{u+s} \cdot \frac{1}{\varrho} = \frac{r}{\varrho}$. Μειώνουμε δηλαδή κατά μία δύναμη το τρέχων r για να πάρουμε το j.

Επιπλέον

$$\begin{aligned} n &= k - j \Rightarrow \\ k &= j + n \end{aligned}$$

και επειδή

$$\begin{aligned} k &\leq r \Rightarrow \\ k &= n + j \leq \lceil \varrho^{u+s} \rceil \end{aligned} \tag{3.8}$$

Έστω $\bar{s}$ ο μικρότερος ακέραιος τέτοιος ώστε $\bar{s} \geq 1$ και

$$\varrho^{u+\bar{s}-1} \geq \max\left(m, \frac{n+1}{\varrho - 1}\right) \tag{3.9}$$

Επιπρόσθετα,

$$r - j = \lceil \varrho^{u+\bar{s}} \rceil - \lceil \varrho^{u+\bar{s}-1} \rceil \geq \varrho^{u+\bar{s}} - \varrho^{u+\bar{s}-1} - 1 = r - j - 1 \geq n$$

κι αυτό γιατί

$$\begin{aligned} k = j + n < r \Rightarrow \\ j + n < r \Rightarrow \\ n < r - j \end{aligned}$$

άρα $s \leq \bar{s}$. Συνεπώς,

$$j \leq \varrho \max\left(m, \frac{n+1}{\varrho - 1}\right) \tag{3.10}$$

και ο αριθμός των βημάτων του αλγόριθμου δίνεται από

$$W_\varrho = k = j + n =\leq \varrho \max\left(m, \frac{n+1}{\varrho - 1}\right) + n \tag{3.11}$$

Εάν $2 \leq \varrho \leq 3$, από την (3.11) παίρνουμε

$$W_\varrho \leq 3(m + n) + 2$$

όπου είναι περίπου ίσο με το όριο του αλγορίθμου του Floyd, το οποίο παρουσιάσαμε στην (3.6).

Η (3.11) αντιστοιχεί στη γενική περίπτωση του B_ϱ, η οποία όμως δεν εγγυάται ότι θα είναι πάντα καλύτερος του Floyd. Για το λόγο αυτό διαλέγουμε μία ειδική περίπτωση με $\varrho = 2$ και $u = 0$ για να κάνουμε την σύγκριση των δύο αλγορίθμων. Επιπλέον τροποποιούμε την (3.9) ως :

$$\varrho^{u+\bar{s}-1} \geq \max\left(m, \frac{n}{\varrho - 1}\right)$$

Η τροποποίηση αυτή είναι εφικτή διότι για $\varrho = 2$, ο παρονομαστής γίνεται μονάδα. Συνοψίζοντας λοιπόν, έχουμε :

$$W_\varrho \leq 2\max(m, n) + n$$

Διακρίνουμε τις περιπτώσεις :

- Για $n > m$, $W_\varrho \leq 2n + n = 3n$. Επιπλέον από την (3.6) έχουμε $3n \leq W_F$. Οπότε $W_\varrho \leq 3n \leq W_F$.
- $m > n$, $W_\varrho \leq 2m + n$ και από την (3.6) έχουμε $3m \leq W_F$. Συνεπώς $W_\varrho \leq 2m + n \leq 3m \leq W_F$.

Καταλήγουμε λοιπόν στο συμπέρασμα ότι ο B_2 είναι πάντα πιο γρήγορος από τον αλγόριθμο του Floyd, αφού ισχύει :

$$\boxed{W_\varrho \leq 2\max(m, n) + n \leq W_F} \tag{3.12}$$

3.4 Αλγόριθμος εύρεσης σύγκρουσης του Sedgewick

Κάθε αλγόριθμος για το πρόβλημα εύρεσης κύκλου πρέπει να έχει χρόνο εκτέλεσης, ο οποίος να υπερβαίνει την ποσότητα $n \cdot t_f$, όπου $n = l + c$. Η ποσότητα t_f θεωρείται σταθερά και αναπαριστά τον χρόνο που χρειάζεται για τον υπολογισμό μιας τιμής της f. Θα μπορούσε να έχει κατασκευαστεί ένας αλγόριθμος με χρόνο εκτέλεσης $n \cdot t_f + O(n \cdot \log(n))$ θεωρώντας ότι όλα τα υπολογισμένα στοιχεία έχουν αποθηκευτεί σε μια αποτελεσματική δομή δεδομένων. Ένας τέτοιος αλγόριθμος όμως δεν είναι ικανοποιητικός για δύο κύριους λόγους. Αφενός δεν είναι ρεαλιστικό να υποθέσουμε ότι υπάρχει κάποια ανεξάντλητη μνήμη και αφετέρου η ανάλυση δεν λαμβάνει υπόψιν το σχετικό κόστος υπολογισμού της f καθώς και της σύγκρισης για ισότητα, δύο στοιχείων του πεδίου ορισμού. Θα παρουσιάσουμε ένα πλαίσιο, εντός του οποίου μπορούν να αντιμετωπιστούν τα δύο παραπάνω προβλήματα [8].

Έστω ένας πίνακας (μια συσχετιστική μνήμη) στον οποίο μπορούν να αποθηκευτούν το πολύ M ζεύγη, (y, i), από στοιχεία του πεδίου ορισμού και από ακεραίους. Κάθε στοιχείο του ζεύγους είναι κλειδί, υπό την έννοια ότι σε κάθε αναζήτηση το πολύ ένα ζευγάρι, με δοσμένη τη μία συνιστώσα θα υπάρχει. Έστω t_u ο χρόνος που απαιτείται για να εισάγουμε ή να διαγράψουμε ένα ζευγάρι από τον πίνακα και t_s ο χρόνος που απαιτείται για την αναζήτηση ενός συγκεκριμένου κλειδιού στον πίνακα. Ανάλογα με την υλοποίηση του πίνακα οι χρόνοι t_u και t_s θα μπορούσαν να είναι σταθερές, λογαριθμικές συναρτήσεις του M ή ακόμα και γραμμικές συναρτήσεις του M.

Θα εξετάσουμε τώρα έναν αποδοτικό αλγόριθμο για το πρόβλημα εύρεσης κύκλου. Ο αλγόριθμος αυτός βασίζεται στην ιδέα του να περιορίσουμε τον αριθμό των λειτουργιών που πραγματοποιούνται στον πίνακα, αποφεύγοντας τις λειτουργίες της αποθήκευσης και της αναζήτησης για τις περισσότερες τιμές της συνάρτησης f. Θα πρέπει να εκτελούνται τόσες τέτοιες λειτουργίες όσες χρειάζονται για την εύρεση μιας σύγκρουσης. Για το λόγο αυτό εισάγουμε δύο νέες παραμέτρους, b και g. Στο Σχήμα 3.3 παρουσιάζεται ο αλγόριθμος, ο οποίος αποθηκεύει μία τιμή στον πίνακα κάθε b φορές που θα εκτελεστεί η f και κάνει αναζήτηση μετά από g αποθηκεύσεις.

Είσοδος: Η συνάρτηση $f : S \to S$, ένα $x_0 \in S$, το πρώτο στοιχείο της ακολουθίας x_j, καθώς και οι παράμετροι $b, g \in \mathfrak{R}$.
Έξοδος: Το ζεύγος (y, i).

1. $i = 0$
2. $y = x$
3. **Επανέλαβε:**
4. **Εάν** $i \equiv 0 \pmod{b}$ **τότε** εισήγαγε (y, i)
5. $y = f(y)$
6. $i = i + 1$
7. **Εάν** $i < b$ $(mod\ gb)$ **τότε** αναζήτησε (y, j)
8. **Μέχρις ότου: βρέθηκε**.

Σχήμα 3.3: Αλγόριθμος εύρεσης σύγκρουσης του Sedgewick.

Τα βήματα αρχικοποίησης μας είναι τα $1, 2$. Στο βήμα 4 ο αλγόριθμος τοποθετεί το ζεύγος (y, i) στον πίνακα χωρίς να εξετάζει αν υπάρχει άλλο ζεύγος με το y ως πρώτη συνιστώσα. Το βήμα 7 μας λέει πως αν το τρέχων y δε βρίσκεται στον πίνακά μας, τότε η μεταβλητή **βρέθηκε** στο βήμα 8 είναι ψευδής, συνεπώς συνεχίζουμε την αναζήτηση επαναλαμβάνοντας τα βήματα $4 - 6$. Αν η αναζήτηση είναι επιτυχής τότε το j είναι η ελάχιστη τιμή για την οποία συναντάμε το ζεύγος (y, j) στον πίνακα.

Παράδειγμα 3.4.1 *Ας δούμε μία εφαρμογή του παραπάνω αλγορίθμου. Έστω* $x = x_0, b = 5$ *και* $g = 7$.

Υπολογίζουμε την ποσότητα : $b \mod (g \cdot b) = 5 \mod (7 \cdot 5) = 5 \mod (35) = 5$.

- $i_0 = 0, y_0 = x_0$
- *Εισαγωγή του* $(y_0, 0)$ *στον πίνακα.*
- $i_1 = 1, y_1 = f(x_0)$. *Εξετάζουμε τώρα αν* $i_1 = 1 = 0 \pmod 5$. *Παρατηρούμε ότι δεν ισχύει, οπότε εξετάζουμε αν* $i_1 = 1 < 5$, *το οποίο ισχύει οπότε αναζητούμε λύση το* y_1 *στον πίνακα.*
- $i_2 = 2, y_2 = f(y_1), i_2 \neq 0 \pmod 5$ *και* $i_2 < 5$, *οπότε συνεχίζουμε την αναζήτηση.*
- $i_3 = 3, y_3 = f(y_2), i_3 \neq 0 \pmod 5$ *και* $i_3 < 5$. *οπότε συνεχίζουμε την αναζήτηση.*
- $i_4 = 4, y_4 = f(y_3), i_4 \neq 0 \pmod 5$ *και* $i_4 < 5$. *οπότε συνεχίζουμε την αναζήτηση.*
- $i_5 = 5, y_5 = f(y_4), i_5 = 0 \pmod 5$. *Εισαγωγή του* $(y_5, 5)$ *στον πίνακα.*
- ….

Είναι δυνατόν ο αλγόριθμός να προσπεράσει το σημείο στο οποίο ο κύκλος κλείνει για πρώτη φορά, αλλά θα το εντοπίσει προτού εκτελεστεί $(n + g \cdot b)$ φορές. Για το λόγο αυτό ο χρόνος εκτέλεσής του φράσσεται από την ποσότητα :

$$(n + g \cdot b) \cdot \left(t_f + \frac{t_s}{g} + \frac{t_u}{b} \right).$$

Σε πολλές στρατηγικές αναζήτησης, η εισαγωγή εφαρμόζεται αφού πρώτα γίνει αναζήτηση της ποσότητας που πρόκειται να γίνει εισαγωγή, με αποτέλεσμα ο αλγόριθμος, να εκτελεί μία μόνο λειτουργία στον πίνακα. Ωστόσο η διαχείρηση της μνήμης και η ανάλυση γίνεται αρκετά περίπλοκη στην περίπτωση αυτή, οπότε προτιμάμε να κάνουμε ξεχωριστά τις αναζητήσεις από τις εισαγωγές. Με την κατάλληλη επιλογή των b και g μπορούμε να μειώσουμε το χρόνο εκτέλεσης του αλγορίθμου. Ο αλγόριθμος κάνει χρήση του ακόλουθου μηχανισμού διαχείρησης μνήμης. Όταν ο πίνακας γεμίσει, αφαιρούνται εκείνα τα ζεύγη (y, j) όπου $\frac{j}{b}$ είναι περιττός ενώ το b διπλασιάζεται. Είναι δανερό πλέον πως το b είναι μεταβλητή ενώ το g παραμένει παράμετρος. Η παραπάνω διαδικασία έχει το ίδιο αποτέλεσμα με το να επανεκκινήσουμε το πρόγραμμα δίνοντας μεγαλύτερη τιμή στο b, χωρίς να είναι αναγκαίες επιπρόσθετες λειτουργίες. Η τελική μορφή του αλγορίθμου παρουσιάζεται στο Σχήμα 3.4.

Επιπλέον η διαχείριση της μνήμης ελέγχεται από μία μεταβλητή, m, η οποία μετράει τον αριθμό των στοιχείων που είναι ήδη στον πίνακα, καθώς και από μία διαδικασία ¨καθαρισμού¨, η οποία απομακρύνει από τον πίνακα όλες τις καταχωρήσεις (z, j), με $\frac{j}{b}$ περιττό.

Είσοδος: Η συνάρτηση $f : S \to S$, το πρώτο στοιχείο $x_0 \in S$ της ακολουθίας, και οι παράμετροι $g, M \in \mathfrak{R}$.
Έξοδος: Το ζεύγος (y, i).

1. $i = 0$
2. $y = x$
3. $b = 1$
4. $m = 0$
5. **Επανέλαβε:**
6. **Εάν** $i \equiv 0 \pmod{b}$ και $m = M$ **τότε**
7. **άρχισε**
8. Εκκαθάριση(b)
9. $b = 2b$
10. $m = \frac{m}{2}$
11. **τελείωσε**
12. **Εάν** $i \equiv 0 \pmod{b}$ **τότε**
13. **άρχισε**
14. **εισήγαγε** (y, i)
15. $m = m + 1$
16. **τελείωσε**
17. $y = f(y)$
18. $i = i + 1$
19. **Εάν** $i < b \mod (g \cdot b)$ **τότε** αναζήτησε (y, j)
20. **Μέχρις ότου: Βρέθηκε**.

Σχήμα 3.4: Τελική έκδοση του αλγόριθμου εύρεσης σύγκρουσης του Sedgewick.

Τα βήματα $6-11$ μας λένε πως αν το i είναι πολλαπλάσιο του b και η μεταβλητή m πάρει τη μέγιστη τιμή της M, γίνει δηλαδή ίση με το μέγιστο αριθμό ζευγαριών που μπορούν να αποθηκευτούν στον πίνακα, τότε ξεκινάει η διαδικασία απομάκρυνσης των υπαρχουσών καταχωρήσεων. Στην περίπτωση αυτή το b διπλασιάζεται και η μεταβλητή m μειώνεται κατά το ήμισυ. Τα βήματα $12-16$ αν $i = k \cdot b$, τότε εισάγουμε το ζεύγος (y, i) με το m να αυξάνεται κατά μία μονάδα. Στα βήματα $17-18$ υπολογίζεται η επόμενη τιμή της ακολουθίας και αυξάνεται κατά 1 η μεταβλητή i. Αν $i < b \mod (g \cdot b)$ αναζητούμε το y, στον πίνακα. Αν βρεθεί επιστρέφεται ο δείκτης j του όρου της ακολουθίας.

Ο αλγόριθμος του σχήματος 3.4 παρουσιάζει κάποιες διαφορές στην εκτέλεσή του από αυτόν του σχήματος 3.3. Ο πρώτος αλγόριθμος "ετρεχε" από την αρχή με την τελική τιμή του b σε αντίθεση με τον τελευταίο στον οποίο οι διαδικασίες καθαρισμού, εισαγωγής και αναζήτησης αλληλεπιδρούν κατάλληλα. Για να λειτουργήσει σωστά ο παραπάνω αλγόριθμος χρειάζεται να τεθούν κάποιοι περιορισμοί στην επιλογή του g, οι οποίοι αναφέρονται στα ακόλουθα λήμματα.

Λήμμα 3.4.0.1 *Εάν το M είναι ένα πολλαπλάσιο του $2g$, τότε η τιμή του i, κατά τη διαδικασία ¨καθαρισμού¨, ικανοποιεί τη σχέση $i \equiv 0 \mod (2 \cdot g \cdot b)$ και $i = 2^k \cdot M$ με $k \geq 0$.*

Απόδειξη 3.4.0.2 *Οι αρχικές τιμές του αλγορίθμου μας είναι $m = 0, b = 1$ και $i = 0$. Το πρώτο ¨εάν¨ θα εκτελεστεί όταν θα έχει γεμίσει η μνήμη, όταν δηλαδή $m = M$ και όταν $i = \lambda b$. Επειδή $b = 1$, ένα στοιχείο για κάθε ομάδα από b στοιχεία προστίθεται στον πίνακα και κάθε φορά που αδειάζει η μνήμη από στοιχεία $b = 2b$, έπεται ότι το $k + 1$-στο άδειασμα θα συμβεί όταν $i = 2^k M$ και $b = 2^{k+1} b$.*

Συνεπώς εξάγουμε το συμπέρασμα ότι κάθε φορά που γίνεται ένα purge, έχουμε $i = b \cdot M$. Επιπλέον αφού το M είναι πολλαπλάσιο του $2 \cdot g$, τότε το i είναι πολλαπλάσιο του $2 \cdot g \cdot b$.

Λήμμα 3.4.0.2 *Η τιμή του b κατά τη διαδικασία της αναζήτησης, δίνεται από τη στρογγυλοποίηση του $\frac{i}{M}$, στην επόμενη δύναμη του 2.*

Απόδειξη 3.4.0.3 *Για να ξεκινήσει ένα look up, πρέπει να έχουμε $i < b \mod (g \cdot b)$, δηλαδή το υπόλοιπο της διαίρεσης του i με το $g \cdot b$, να είναι μικρότερο του b.*

Από το προηγούμενο λήμμα έχουμε $i = b \cdot M$ κάθε φορά που εκτελείται ένα purge. Επίσης, είδαμε ότι μετά από $k + 1$ purges, το b παίρνει την τιμή 2^{k+1}. Οπότε μετά από $k + 1$ purges θα έχουμε ξανά διαγραφή στοιχείων όταν $i = 2^{k+1} \cdot M$.

Συνεπώς οι τιμές του i κυμαίνονται μεταξύ $2^k \cdot M$ και $2^{k+1} \cdot M$. Δηλαδή ισχύει η παρακάτω σχέση :

$$2^k \cdot M < i \leq 2^{k+1} \cdot M. \tag{3.13}$$

Γνωρίζουμε πως $b = \frac{i}{M}$ και διαιρώντας την (3.13) με M παίρνουμε την παρακάτω ανισότητα :

$$2^k < b \leq 2^{k+1} \tag{3.14}$$

Με τη σχέση (3.14) τελειώνει και η απόδειξη του λήμματος 3.4.0.2.

Λήμμα 3.4.0.3 *Κατά τη διάρκεια της εκτέλεσης του αλγορίθμου εύρεσης σύγκρουσης, όποτε γίνεται αναζήτηση, εισαγωγή και εκκαθάριση στοιχείων του πίνακα, έχουμε ότι $(f^j(x), j)$ βρίσκεται στον πίνακα αν και μόνο αν $0 \leq j < i$ και $j \equiv 0 \pmod b$.*

Απόδειξη 3.4.0.4 *Από τον ορισμό της διαδικασίας εκκαθάρισης έχουμε ότι σε κάθε εκτέλεση του purge, απομακρύνονται από τον πίνακα όλα τα ζεύγη (y, j) με $\frac{j}{b}$ περιττό. Συνεπώς παραμένουν στον πίνακα μόνο τα ζεύγη (y, j) με $\frac{j}{b}$ άρτιο, δηλαδή $j = \lambda \cdot b, \lambda$ άρτιος.*

Άρα από τη σχέση (3.13) και από το ότι το j διαιρείται ακριβώς με το b παίρνουμε τις ζητούμενες σχέσεις :

$$0 \leq j < i, j \equiv 0 \pmod{b}. \tag{3.15}$$

Λήμμα 3.4.0.4 *Εάν το M είναι ένα πολλαπλάσιο του $2g$, τότε ο αλγόριθμος εύρεσης κύκλου τερματίζει όταν :*

$$n \leq i < n + (g+2) \cdot b_n,$$

όπου b_n υπολογίζεται από τη στρογγυλοποίηση του $\frac{n}{M}$, στην επόμενη δύναμη του 2.

Απόδειξη 3.4.0.5 *Δεδομένου ότι στον τερματισμό του αλγορίθμου πρέπει να έχουμε $i > j$ και $f^i(x) = f^j(x)$, είναι σαφές από τον ορισμό της περιόδου, $n = l + c$, ότι το i δε μπορεί να είναι μικρότερο του n. Για να ολοκληρώσουμε την απόδειξη, χρειάζεται να δείξουμε ακόμα ότι $i < n + (g+2) \cdot b_n$.*

Υποθέτουμε ότι ο αλγόριθμός μας συνεχίζει να ¨τρέχει¨, όταν $i = n + (g+2) \cdot b_n$. Έστω i_0, ο μοναδικός ακέραιος για τον οποίο ισχύουν τα παρακάτω:

$$n \leq i_0 < n + g \cdot b_n$$

και

$$i_0 \equiv 0 \pmod{g \cdot b_n}.$$

Τότε το i_0 είναι η πρώτη τιμή που παίρνει το i, όταν ξεπεράσει το n, για το οποίο ο αλγόριθμός μας θα κάνει αναζήτηση σε ένα τμήμα απο b_n διαδοχικές τιμές. Από το λήμμα 3.4.0.1, το i_0 είναι η πρώτη τιμή που παίρνει το i, όταν ξεπεράσει το n, για το οποίο αρχίζει η διαδικασία της εκκαθάρισης. Έτσι προκύπτουν δύο περιπτώσεις.

- *Έστω $i_0 \neq 2^k \cdot M$. Από το λήμμα 3.4.0.1 γνωρίζουμε ότι δεν πρόκειται να ακολουθήσει η διαδικασία της εκκαθάρισης, καθ΄ όλη τη διάρκεια των b_n αναζητήσεων, όταν το i γίνει ίσο με i_0. Επιπλέον αφού $i_0 \geq n$, οι τιμές που προκύπτουν από τις αναζητήσεις είναι ίδιες με τις b_n τιμές, όταν $i = i_0 - c$. Βάσει του λήμματος 3, μία από αυτές τις τιμές πρέπει να υπάρχει στον πίνακα. Έτσι ο αλγόριθμος τερματίζει με $i \leq i_0 + b_n < n + (g+1) \cdot b_n$, το οποίο έρχεται σε αντίφαση με την υπόθεσή μας, οπότε καταλήγουμε σε άτοπο.*

- *Έστω $i_0 = 2^k \cdot M$. Από το λήμμα 1 εξάγουμε το συμπέρασμα ότι μετά την αναζήτηση του $f^{i_0}(x)$, θα αρχίσει η διαδικασία της εκκαθάρισης, καθώς και ότι $i_0 \equiv 0 \pmod{2 \cdot g \cdot b_n}$. Άρα μετά την εκκαθάριση θα έχουμε $b = 2 \cdot b_n$ και $i_0 \equiv 0 \pmod{g \cdot b}$. Συνεπώς όταν $i = i_0$, ο αλγόριθμός μας αρχίζει τις αναζητήσεις στις $2 \cdot b_n$ τιμές του πίνακα. Όπως και προηγουμένως μία από αυτές τις τιμές πρέπει να υπάρχει στον πίνακα. Ο αλγόριθμός μας επομένως τερματίζει σε $i \leq i_0 + 2 \cdot b_n < n + (g+2) \cdot b_n$, το οποίο πάλι καταλήγει σε άτοπο.*

Η συνθήκη ότι το M πρέπει να είναι πολλαπλάσιο του $2 \cdot g$, μας βοηθάει στο να βρούμε ένα όριο για τον αλγόριθμο, το οποίο έχει ουσιαστική σημασία για την ορθή εκτέλεσή του. Εάν δεν ισχύει αυτή η συνθήκη είναι πιθανό να βρεθούν l και c, για τα οποία ο αλγόριθμος μπορεί να ¨κολλήσει¨ σε ένα βρόχο, όπου καμία αναζήτηση δεν θα καταλήγει στην εύρεση του κύκλου, εξαιτίας της προηγούμενης εκκαθάρισης.

Αυτά τα λήμματα περιγράφουν την απόδοση του αλγορίθμου, ώστε να μπορέσουμε να υπολογίσουμε τη χειρότερη χρονική του απόδοση. Διαισθητικά αναμένουμε ότι ο απαιτούμενος χρόνος μειώνεται καθώς αυξάνονται τα g, M. Ένα μικρό διάστημα αναζητήσεων συνεπάγεται συχνές αναζητήσεις, αλλά έχει μικρή πιθανότητα να προσπεράσει την αρχή του κύκλου. Αντιθέτως ένα μεγάλο διάστημα αναζητήσεων περιορίζει τον αριθμό των αναζητήσεων, όμως μεγαλώνουν οι πιθανότητες να υπερβεί κατά πολύ την αρχή του κύκλου.

Πριν προχωρήσουμε στην λεπτομερή ανάλυση που χρειαζόμαστε για να ολοκληρώσουμε τη λύση του προβλήματος, καθορίζουμε τα l, c μόλις ανιχνευθεί ο κύκλος.

Ο αλγόριθμος εύρεσης του κύκλου τερματίζει όταν βρει ένα ζεύγος (i, j), με $i > j$, για το οποίο ισχύει $f^i(x) = f^j(x)$. Αυτό συνεπάγεται ότι $j > l$ και $i \equiv j \pmod{c}$. Για την εύρεση των l και c πρέπει να χρησιμοποιήσουμε τις αποθηκευμένες τιμές στον πίνακα. Παρακάτω παρουσιάζεται ένας αλγόριθμος, ο οποίος ανακτά τη λύση (l, c), κάθε φορά που ο αλγόριθμος εύρεσης κύκλου τεματίζει.

Είσοδος: Η συνάρτηση $f : S \to S$, το πρώτο στοιχείο $x_0 \in S$ της ακολουθίας, καθώς και οι παράμετροι $b, g \in \mathfrak{R}$.
Έξοδος: Το ζεύγος (l, c).

1. $i' = g \cdot b \cdot \lfloor \frac{i}{g \cdot b} \rfloor - g \cdot b$
2. $j' = j - (i - i')$
3. **Εάν** $i' > j$
4. **τότε:** $c = i - j$
5. **διαφορετικά** $c =$ η μικρότερη τιμή του c, ώστε $f^j(x) = f^{j+c}(x)$
6. $l = j' +$ μικρότερη τιμή του l', ώστε $f^{j'+l'}(x) = f^{i'+l'}(x)$

Σχήμα 3.5: Αλγόριθμος ανάκτησης

Στον παραπάνω αλγόριθμο το i' επισημαίνει την αρχή των πρώτων αναζητήσεων. Επιπλέον αφού $i' \equiv 0 \pmod{g \cdot b}$, από το λήμμα 3.4.0.3 γνωρίζουμε ότι το $f^{i'}(x)$ βρίσκεται στον πίνακα. Παρ' όλο που το $f^{j'}(x)$ δεν είναι απαραίτητο να αποθηκευτεί στον πίνακα, μπορεί να βρεθεί κάνοντας μία αναζήτηση για $f^{b \cdot \lfloor \frac{j'}{b} \rfloor}(x)$ και εφαρμόζοντας την f ακριβώς $j' \pmod{b}$ φορές. Αυτό θα γίνει το πολύ σε $t_s + 2 \cdot b_n \cdot t_f$ χρόνο, εξαιτίας του ότι η τελική τιμή του b είναι b_n ή $2 \cdot b_n$.

Λήμμα 3.4.0.5 *Ο αλγόριθμος ανάκτησης βρίσκει σωστά το c.*

Απόδειξη 3.4.0.6 *Αφού* $f^i(x) = f^j(x)$, *έχουμε* $i \equiv j \pmod{c}$, *δηλαδή το μέγεθος του κύκλου,* c, *διαιρεί το* $i - j$. *Εάν* $i' \leq j$, *τότε :*

$$i - j \leq g \cdot b + b \leq (g+1) \cdot 2 \cdot b_n.$$

Εάν $i' > j$, *τότε ο αναζητήσεις για* $f^{i'}(x), ..., f^{i'+b_n-1}$, *εκτελέστηκαν χωρίς επιτυχία, μεταξύ* j *και* i *και ο κύκλος πρέπει να έχει περαστεί μόνο μία φορά. Για τον λόγο αυτό το* c *είναι ακριβώς* $i - j$.

Λήμμα 3.4.0.6 *Ο αλγόριθμος ανάκτησης βρίσκει σωστά το* l.

Απόδειξη 3.4.0.7 *Ξέρουμε πως ισχύουν τα παρακάτω :*

$$j' < l \leq j$$

και

$$i' < l + c \leq i,$$

διαφορρετικά ο αλγόριθμος θα είχε τερματίσει νωρίτερα. Επίσης

$$i \equiv j \pmod{c}$$

και

$$i - i' = j - j'.$$

Άρα $j' \equiv i' \pmod{c}$.

3.4.1 Worst case περίπτωση

Οι αλγόριθμοι του προηγούμενου εδαφίου μπορούν να μας δώσουν μια αποτελεσματική λύση στο πρόβλημα του κύκλου, εάν η παράμετρος g επιλέγεται σωστά. Στην ενότητα αυτή, θα αναλύσουμε το χρόνο λειτουργίας των αλγορίθμων για να κάνουμε την καλύτερη επιλογή του g. Το g αυτό θα πρέπει να ελαχιστοποιεί το χρόνο εκτέλεσης του αλγορίθμου.

Αρχικά θα πρέπει να προσθέσουμε όλα τα κόστη, που προκύπτουν όταν οι αλγόριθμοι ¨τρέχουν¨ για να δώσουν την λύση (l, c), για ένα συγκεκριμένο στιγμιότυπο, (f, x), του προβλήματος εύρεσης κύκλου.

Στη χειρότερη περίπτωση, ο απαιτούμενος χρόνος για να τερματίσει ο αλγόριθμος είναι :

$$n \cdot \left(1 + \frac{2 \cdot g + 4}{M}\right) \cdot \left(t_f + \frac{t_s}{g}\right) + t_u \cdot M \cdot \log_2 \frac{4\sqrt{2}n}{M}$$

Ο επιπρόσθετος χρόνος που απαιτείται για να βρεθούν τα l, c είναι το πολύ :

$$n \cdot \frac{4 \cdot (3 \cdot g + 1)}{M} \cdot t_f + t_s$$

Επιπλεόν εάν το M είναι δύναμη του 2 και εάν το g έχει επιλεχθεί έτσι ώστε να είναι η πλησιέστερη δύναμη του 2 προς $\sqrt{\frac{(M+4)\cdot t_s}{14\cdot t_f}}$, τότε ο συνολικός χρόνος εκτέλεσης του αλγορίθμου είναι λιγότερος από :

$$n \cdot t_f \cdot \left(1 + O\left(\sqrt{\frac{t_s}{M}}\right)\right).$$

Κεφάλαιο 4

Εφαρμογές Αλγορίθμων Εύρεσης Συγκρούσεων στην Κρυπτανάλυση

Όπως έχουμε ήδη αναφέρει στο κεφάλαιο 1, η ασφάλεια του RSA κρυπτοσυστήματος, βασίζεται στη δυσκολία παραγοντοποίησης ενός σύνθετου αριθμού. Στο κεφάλαιο αυτό θα παρουσιάσουμε τον αλγόριθμο του Pollard, βάσει του οποίου μπορούμε να βρούμε έναν τέτοιο παράγοντα. Επίσης θα περιγράψουμε τους αλγορίθμους του Shank και Pollard Rho, με τους οποίους μπορούμε να λύσουμε το πρόβλημα του διακριτού λογαρίθμου, στο οποίο στηρίζεται η ασφάλεια του κρυπτοσυστήματος El Gamal, καθώς και των ελλειπτικών καμπυλών. Η βάση των αλγορίθμων αυτών είναι το πρόβλημα εύρεσης σύγκρουσης ή κύκλου σε μια περιοδική ακολουθία. Για να γίνουν κατανοητοί οι παρακάτω αλγόριθμοι, θα πρέπει πρώτα να δώσουμε κάποιους ορισμούς.

Κυκλική ομάδα καλείται μια ομάδα G, αν υπάρχει στοιχείο $\alpha \in G$, τέτοιο ώστε $G =< \alpha >$, δηλαδή $\forall x \in G, \exists y : x = \alpha^y$. Το στοιχείο α, το ονομάζουμε **γεννήτορα** της G. **Τάξη** μίας κυκλικής ομάδας, $G =< \alpha >$, ονομάζεται ο μικρότερος ακέραιος, n, τέτοιος ώστε $\alpha^n = c$. Τότε $G =< \alpha >= \{\alpha^0, ..., \alpha^{n-1}\}$ και $|G| = n$.

Έστω G, μία πεπερασμένη κυκλική ομάδα, τάξης n, α ένας γεννήτορας της και $\beta \in G$. Ο **διακριτός λογάριθμος του β ως προς τη βάση α**, $\log_\alpha(\beta)$, είναι ο μοναδικός ακέραιος x, με $0 \leq x \leq n-1$, τέτοιος ώστε $\beta = \alpha^x$.

Το **πρόβλημα διακριτού λογαρίθμου**, (DLP), είναι το εξής: δοθέντος ενός πρώτου αριθμού p, ενός γεννήτορα α του Z_p και ενός στοιχείου $\beta \in Z_p$, να βρεθεί ο ακέραιος $x, 0 \leq x \leq p-2$, τέτοιος ώστε $\alpha^x \equiv \beta \pmod{p}$.

Το **γενικευμένο πρόβλημα διακριτού λογαρίθμου**, (GDLP), είναι το εξής: δοθείσης μιας πεπερασμένης κυκλικής ομάδας G τάξης n, ενός γεννήτορα α της G και ενός στοιχείου $\beta \in G$, να βρεθεί ο ακέραιος $x, 0 \leq x \leq n-1$, τέτοιος ώστε $\alpha^x = \beta$.

4.1 Ο Αλγόριθμος Παραγοντοποίησης Ακεραίων του Pollard

Θα μελετήσουμε τον Monte Carlo αλγόριθμο παραγοντοποίησης του Pollard, με τον οποίο μπορούμε να βρούμε έναν παράγοντα ενός σύνθετου αριθμού. Ο αλγόριθμος του Pollard βασίζεται στον αλγόριθμο του Floyd, ο οποίος βρίσκει την περίοδο n μιας ακολουθίας x_j.

4.1.1 Ο Αλγόριθμος του Pollard βάσει του αλγόριθμου του Floyd

Ο Pollard πρότεινε να αντικατασταθεί η $f(x)$ στον αλγόριθμο του Floyd από ένα κατάλληλο πολυώνυμο $\bmod N$, καθώς και το κριτήριο τερματισμού, $x = y$, να γίνει Μ.Κ.Δ.$(|x-y|, N) > 1$. Συνήθως επιλέγουμε $f(x) = x^2 \pm 1$ και ο αλγόριθμός μας τερματίζει όταν το $|x-y|$ διαιρεί το N, όπου N ο σύνθετος αριθμός που θέλουμε να παραγοντοποιήσουμε, $x = x_j$ και $y = x_{2j}$ [7].

Έστω p ο μικρότερος πρώτος παράγοντας του N και :

$$\overline{x_i} = x_i \pmod{p}$$

Επειδή η $f(x)$ είναι πολυωνυμική και $p|N$, με τη βοήθεια της (3.1), θα ισχύει :

$$\overline{x_{i+1}} = f(x_i) \pmod{p} \tag{4.1}$$

Στη συνέχεια παράγουμε τις τριπλέτες :

$$(x_i, x_{2i}, Q_i), i = 1, 2, ...$$

όπου

$$Q_i = \prod_{j=1}^{i} (x_{2j} - x_j) \pmod{N} \tag{4.2}$$

Κάθε τριπλέτα παράγεται από 3 εφαρμογές της (4.1) και ένα πολλαπλασιασμό στην (4.2). Στην ουσία λοιπόν έχουμε 4 πολλαπλασιασμούς $(\bmod N)$. Χρησιμοποιούμε τέσσερις μεταβλητές, x_i, x_{2i}, Q_i, N. Όταν $i = k \cdot m, k = 1, 2, ...$ υπολογίζουμε το Μ.Κ.Δ.$(Q_i, N) = d_i$.

Παρακάτω δίνεται ο αλγόριθμος του Pollard.

Τα βήματα αρχικοποίησης είναι τα $1-3$ και τα βήματα $5-9$ είναι τα ίδια με του Floyd, με τον επιπλέον υπολογισμό του Q και του d_i. Για τον υπολογισμό του d_i διακρίνουμε τις περιπτώσεις :

Είσοδος: Μία κατάλληλη πολυωνυμική συνάρτηση $f \pmod N$, ένα x_0 και ένας σύνθετος αριθμός N.
Έξοδος: Ένας παράγοντας του N, d_i, τέτοιος ώστε : $d_i = \text{Μ.Κ.Δ.}(Q_i, N) > 1$

1. $x := x_0$
2. $y := x_0$
3. $Q := 1$
4. **Επανέλαβε:**
5. $\quad x := f(x)$
6. $\quad y := f(x)$
7. $\quad y := f(y)$
8. $\quad Q := Q \cdot (y - x)$
9. $\quad d_i = \text{Μ.Κ.Δ}(Q, N) > 1$
10. **Μέχρις ότου:** $d_i \neq 1$

Σχήμα 4.1: Ο αλγόριθμος εύρεσης συγκρούσεων του Pollard.

- $d_i = N$
- $d_i = 1$
- $1 < d_i < N$

Στη πρώτη περίπτωση δεν εξάγεται κάποιο συμπέρασμα, οπότε επιλέγουμε διαφορετικό x_0 ή ακόμα και άλλη $f(x)$. Στη δεύτερη περίπτωση συνεχίζουμε υπολογίζοντας το επόμενο $Q_i = Q_{i-1} \cdot (x_{2i} - x_i)$ μέχρι να φτάσουμε στη τελευταία, όπου έχουμε καταφέρει μία μερική παραγοντοποίηση του N. Εάν όμως ο d_i είναι σύνθετος, πρέπει να παραγοντοποιηθεί. Επομένως συνεχίζουμε την ίδια διαδικασία βάζοντας $N' = \frac{N}{d_i}$, μέχρις ότου ο d_i να γίνει πρώτος ή να φτάσουμε σε ένα μέγιστο προκαθορισμένο αριθμό βημάτων, S'.

Παράδειγμα 4.1.1 *Έστω ότι έχουμε $f(x) = x^2 + 1$ με $x_0 = 2$ και θέλουμε να παραγοντοποιήσουμε το σύνθετο αριθμό $N = 7171$.*

Αρχίζουμε να παράγουμε τις τριπλέτες (x_j, x_{2j}, Q_i).

Για $i = 1$ έχουμε :

- $x_1 = f(x_0) = 2^2 + 1 = 5$
- $y_1 = f(x_1) = f(5) = 26$
- $Q_1 = (y_1 - x_1) \bmod 7171 = (26 - 5) \bmod 7171 = 19$

Ελέγχουμε τώρα αν $d_1 > 1$. Μ.Κ.Δ.$(Q_1, 7171) =$ Μ.Κ.Δ.$(19, 7171) = 1 = d_1$.

Οπότε συνεχίζουμε την παραγωγή τριπλετών μέχρι να βρούμε i *τέτοιο ώστε* $d_i > 1$.

Για $i = 2$ *έχουμε :*

- $x_2 = f(x_1) = f(26) = 677$
- $y_2 = f(x_2) = f(677) = 458330 \bmod 7171 = 6557$
- $Q_2 = Q_1 \cdot (y_2 - x_2) \bmod 7171 = 19 \cdot 6531 \bmod 7171 = 124089 \bmod 7171 = 2182$

$d_2 = M.K.\Delta.(2182, 7171) = 1$

Για $i = 3$ *έχουμε :*

- $x_3 = f(x_2) = f(6557) = 42994250 \bmod 7171 = 4105$
- $y_3 = f(x_3) = f(4105) = 16851026 \bmod 7171 = 6347$
- $Q_3 = Q_2 \cdot (y_3 - x_3) \bmod 7171 = 2182 \cdot 5670 \bmod 7171 = 12371940 \bmod 7171 = 1965$

$d_3 = M.K.\Delta.(1965, 7171) = 1$

Για $i = 4$ *έχουμε :*

- $x_4 = f(x_3) = f(6347) = 40284410 \bmod 7171 = 4903$
- $y_4 = f(x_4) = f(4903) = 24039410 \bmod 7171 = 2218$
- $Q_4 = Q_3 \cdot (y_4 - x_4) \bmod 7171 = 1965 \cdot (-4339) \bmod 7171 = 1965 \cdot 2832 \bmod 7171 = 5564880 \bmod 7171 = 184$

$d_4 = M.K.\Delta.(184, 7171) = 1$

Για $i = 5$ *έχουμε :*

- $x_5 = f(x_4) = f(2218) = 4919525 \bmod 7171 = 219$
- $y_5 = f(x_5) = f(219) = 47962 \bmod 7171 = 4936$
- $Q_5 = Q_4 \cdot (y_5 - x_5) \bmod 7171 = 184 \cdot 831 \bmod 7171 = 152904 \bmod 7171 = 2313$

$d_5 = M.K.\Delta(2313, 7171) = 1$

Για $i = 6$ *έχουμε :*

- $x_6 = f(x_5) = f(4936) = 24364097 \bmod 7171 = 4210$

- $y_6 = f(x_6) = f(4210) = 17724101 \bmod 7171 = 4560$
- $Q_6 = Q_5 \cdot (y_6 - x_6) \bmod 7171 = 2313 \cdot (-1787) \bmod 7171 = 2313 \cdot 5384 \bmod 7171 = 12453192 \bmod 7171 = 4336$

$d_6 = M.K.\Delta.(4336, 7171) = 1$

Για $i = 7$ *έχουμε :*

- $x_7 = f(x_6) = f(4560) = 20793601 \bmod 7171 = 4872$
- $y_7 = f(x_7) = f(4872) = 23736385 \bmod 7171 = 375$
- $Q_7 = Q_6 \cdot (y_7 - x_7) \bmod 7171 = 4336 \cdot (-4528) \bmod 7171 = 4336 \cdot 2643 \bmod 7171 = 11460048 \bmod 7171 = 790$

$d_7 = M.K.\Delta.(790, 7171) = 1$

Για $i = 8$ *έχουμε :*

- $x_8 = f(x_7) = f(375) = 140626 \bmod 7171 = 4377$
- $y_8 = f(x_8) = f(4377) = 19158130 \bmod 7171 = 4389$
- $Q_8 = Q_7 \cdot (y_8 - x_8) \bmod 7171 = 790 \cdot 2171 \bmod 7171 = 1715090 \bmod 7171 = 1221$

$d_8 = M.K.\Delta.(1221, 7171) = 1$

Για $i = 9$ *έχουμε :*

- $x_9 = f(x_8) = f(4389) = 19263322 \bmod 7171 = 2016$
- $y_9 = f(x_9) = f(2016) = 4064257 \bmod 7171 = 5471$
- $Q_9 = Q_8 \cdot (y_9 - x_9) \bmod 7171 = 1221 \cdot 5252 \bmod 7171 = 6412692 \bmod 7171 = 1818$

$d_9 = M.K.\Delta.(1818, 7171) = 101$

Παρατηρούμε ότι ο 101 *είναι πρώτος αριθμός οπότε έχουμε βρει έναν πρώτο παράγοντα του* 7171. *Συνεπώς* $7171 = 101 \cdot 71$,*με* $N = 7171, p = 101, q = 71$ ■

Ας υποθέσουμε τώρα ότι το N στην (4.1) αντικαθίσταται από έναν πρώτο αριμό, έστω p. Η ακολουθία x_i είναι περιοδική, οπότε θα υπάρχουν ακέραιοι $c \geq 1$ και $t \geq 0$ τέτοιοι ώστε $\forall i \geq t$ να ισχύει :

$$x_{c+i} = x_i \pmod{p}.$$

Παρατηρούμε ότι στον Floyd το c αναπαρίσταται από το n και το t από το m.

Συνεχίζουμε ορίζοντας το r να είναι ο ελάχιστος θετικός ακέραιος για τον οποίο ισχύει :

$$x_r = x_{2r} \pmod{p}.$$

Η συνάρτηση $r = r(p)$ καθορίζει το πόσο γρήγορα θα τερματίσει ο αλγόριθμός μας και το r παίρνει τις τιμές :

$$\begin{Bmatrix} t \leq r < t + c, r \equiv 0 \pmod{c}, t > 0 \\ r = c, t = 0 \end{Bmatrix}$$

Περιμένουμε μετά από r βήματα να έχουμε $Q_i = 0 \pmod{p}$ και αυτό γιατί αφού ο αλγόριθμος θα έχει τερματίσει θα έχουμε $d_i = p$ και $d_i = \text{Μ.Κ.Δ.}(Q_i, N)$, οπότε $Q_i = k \cdot p$.

Τέλος κάνουμε τις εξής υποθέσεις :

1. Η $f(x)$ μπορεί να αντικατασταθεί από όλα τα πολυώνυμα της μορφής $x^2 + b$ για κάθε $b \in \mathfrak{R}$ εκτός από $b = 0$ και $b = -2$, ανεξαρτήτως της αρχικής τιμής x_0.

2. Εάν είναι γνωστός ο πρώτος παράγοντας, p, του N, τέτοιος ώστε να ικανοποιείται η $p \equiv 1 \pmod{k}$, με $k > 2$, μπορούμε να αντικαταστήσουμε την $x^2 + b$ από την $x^k + b$ και το p αντικαθίσταται από το $\frac{p}{k-1}$.

4.2 Αλγόριθμοι για το πρόβλημα του διακριτού λογαρίθμου

Σε αυτήν την ενότητα υποθέτουμε ότι $(G, \cdot)$ είναι μία πολλαπλασιαστική ομάδα και το $\alpha \in G$ είναι n τάξεως. Ως εκτούτου το πρόβλημα του διακριτού λογαρίθμου μπορεί να διατυπωθεί ως εξής :

Με δοσμένο $\beta \in < \alpha >$, βρείτε το μοναδικό εκθέτη α, με $0 \leq \alpha \leq n - 1$, τέτοιον ώστε $\alpha^\alpha = \beta$. [5]

Ξεκινάμε με την ανάλυση στοιχειωδών αλγορίθμων οι οποίοι μπορούν να λύσουν το πρόβλημα του διακριτού λογαρίθμου. Θα υποθέσουμε πως ο χρόνος που απαιτείται για να υπολογίσουμε το γινόμενο δύο στοιχείων στην ομάδα G, είναι σταθερός.

Αρχικά παρατηρούμε ότι το πρόβλημα του διακριτού λαγαρίθμου, μπορεί να λυθεί με εξαντλητική αναζήτηση, σε $O(n)$ χρόνο και $O(1)$ χώρο, υπολογίζοντας τις δυνάμεις του α, δηλαδή : $\alpha, \alpha^2, \alpha^3, ...$, μέχρι να βρεθεί ένα β, τέτοιο ώστε $\beta = \alpha^\alpha$. Κάθε α^i

δύναμη του α, υπολογίζεται πολλαπλασιάζοντας τη προηγούμενη δύναμη α^{i-1}, με α, για το λόγο αυτό ο χρόνος υπολογισμού είναι $O(n)$.

Μία άλλη προσέγγιση του προβλήματος είναι να υπολογίσουμε όλες τις δυνάμεις α^i και έπειτα να ταξινομήσουμε όλα τα ζεύγη (i, α^i), βάσει της δεύτερης συντεταγμένης τους. Στη συνέχεια με δοσμένο β, μπορούμε να κάνουμε μία αρχική αναζήτηση της τιμής του α, ώστε $\beta = \alpha^\alpha$, στη ταξινομημένη λίστα. Αυτή η αναζήτηση απαιτεί $O(n)$ χρόνο, αφού το πλήθος των δυνάμεων του α είναι n, καθώς και $O(n \cdot \log n)$ χρόνο, για να ταξινομηθεί η λίστα n στοιχείων. Εάν αγνοήσουμε τους λογαριθμικούς παράγοντες, όπως γίνεται συνήθως στην ανάλυση τέτοιων αλγορίθμων, ο χρόνος υπολογισμού θα είναι $O(n)$. Ο απαιτούμενος χρόνος για την αρχική αναζήτηση της ταξινομημένης λίστας, n μεγέθους, είναι $O(\log n)$. Εάν πάλι παραλήψουμε τους λογαριθμικούς όρους, τότε μπορούμε να λύσουμε το πρόβλημα του διακριτού λογαρίθμου σε σταθερό χρόνο, $O(1)$, με $O(n)$ υπολογισμούς και $O(n)$ απαιτούμενη μνήμη.

4.2.1 Ο αλγόριθμος του Shank

Είσοδος: Μία κυκλική ομάδα G, ένας γεννήτορας $\alpha \in G$, τάξης n και ένα $\beta \in < \alpha >$.
Έξοδος: Ο διακριτός λογάριθμος $\log_\alpha(\beta)$.

1. $m = \lceil \sqrt{n} \rceil$
2. **Για:** $0 \leq j \leq m-1$
3. **κάνε** υπολογισμό του $\alpha^{m \cdot j}$
4. Ταξινόμησε τα m πλήθους ζεύγη, $(j, \alpha^{m \cdot j})$, βάσει της δεύτερης συνιστώσας τους, δημιουργώντας τη λίστα L_1.
5. **Για** $0 \leq i \leq m-1$
6. **κάνε** υπολογισμό του $\beta \cdot \alpha^{-i}$
7. Ταξινόμησε τα m πλήθους ζεύγη, $(i, \beta \cdot \alpha^{-i})$, βάσει της δεύτερης συνιστώσας τους, δημιουργώντας τη λίστα L_2.
8. Βρες ένα ζευγάρι, $(j, y) \in L_1$ και ένα ζευγάρι, $(i, y) \in L_2$.
9. $\log_\alpha \beta = (m \cdot j + i) \pmod n$

Σχήμα 4.2: Ο αλγόριθμος ανίχνευσης του Shank.

Από το βήμα 8 εξάγουμε το συμπέρασμα ότι :

$$\alpha^{m \cdot j} = y = \beta \cdot \alpha^{-i}$$

και επομένως έχουμε το ζητούμενο :

$$\alpha^{m \cdot j + i} = \beta. \tag{4.3}$$

Αντιστρόφως, για κάθε $\beta \in < \alpha >$, έχουμε :

$$0 \leq \log_\alpha \beta \leq n-1$$

Αν τώρα λογαριθμίσουμε την (4.3) παίρνουμε :

$$\log_\alpha \beta = m \cdot j + i,$$

όπου $0 \leq j \leq m-1, 0 \leq i \leq m-1$.

Ξέρουμε ότι :

$$\log_\alpha \beta \leq n-1 \leq m^2 - 1 = m \cdot (m-1) + m - 1 \tag{4.4}$$

επίσης

$$m \cdot j + i = \log_\alpha \beta \tag{4.5}$$

Από τις παραπάνω σχέσεις (4.4) και (4.5), έχουμε ότι :

$$m \cdot j + i \leq m \cdot (m-1) + m - 1$$

και εάν $i = m-1$, τότε :

$$m \cdot j + (m-1) \leq m \cdot (m-1) + m - 1$$

και καταλήγουμε :

$$j \leq m-1. \tag{4.6}$$

Τόσο ο χρόνος υπολογισμού, όσο και η απαιτούμενη μνήμη, είναι της τάξεως $O(m)$, αγνωώντας τους λογαριθμικούς παράγοντες. Στα βήματα $2,3$, μπορούμε να υπολογίσουμε πρώτα το α^m και έπειτα να το πολλαπλασιάζουμε με τις δυνάμεις του α. Ο συνολικός χρόνος υπολογισμού αυτών των βημάτων είναι $O(m)$, αφού έχουμε m πλήθος δυνάμεις του α. Αντίστοιχα και για τα βήματα $5,6$, ο συνολικός χρόνος υπολογισμού είναι $O(m)$. Τα βήματα $4,7$ απαιτούν χρόνο τάξεως $O(m \cdot \log m)$. Το βήμα 8 ολοκληρώνεται με τη σύγκριση των δύο λιστών, L_1 και L_2, οπότε ο χρόνος που απαιτείται είναι $O(m)$.

Παράδειγμα 4.2.1 *Ας δώσουμε τώρα ένα παράδειγμα του αλγορίθμου του Shank.*

Ας υποθέσουμε ότι θέλουμε να υπολογίσουμε το $\log_3 525$ *στην ομάδα* $(Z^*_{809}, \cdot)$. *Οπότε* $p = 809$, *το* 3 *είναι η πρωταρχική ρίζα* $\mod 809$, *άρα* $\alpha = 3$, $\beta = 525, n = 808$ *και* $m = \lceil \sqrt{808} \rceil = 29$.

Υπολογίζουμε τώρα το $\alpha^m \pmod p$, *το οποίο είναι :*

$$3^{29} = (\quad \mod 809) = 99$$

Στη συνέχεια υπολογίζουμε τα ζεύγη $(j, 99^j (\mod 809))$, *για* $0 \leq j \leq m-1 = 28$ *και σχηματίζουμε τον πίνακα* L_1, *ο οποίος δίνεται ταξινομημένος παρακάτω.*

$L_1 = \{(0,1), (23,15), (12,26), (28,81), (2,93), (1,99), (13,147), (8,207), (6,211), (9,268), (19,275),$

$(19,275), (27,295), (3,308), (5,329), (17,464), (21,496), (20,528), (4,559), (22,564), (26,575),$

$(25,586), (18,638), (10,644), (11,654), (7,664), (24,676), (15,725), (16,781), (14,800)\}.$

Τώρα θα υπολογίσουμε τα ζεύγη $(i, 525 \cdot (3^i)^{-1} (\mod 809))$, *για* $0 \leq i \leq m-1 = 28$ *και σχηματίζουμε τον πίνακα* L_2, *ο οποίος δίνεται ταξινομημένος παρακάτω.*

$L_2 = \{(6,44), (5,132), (17,133), (28,163), (1,175), (13,256), (24,259), (18,314), (2,328), (14,355),$

$(25,356), (3,379), (15,388), (4,396), (16,399), (10,440), (27,489), (9,511), (21,521), (0,525),$

$(7,554), (19,644), (26,658), (11,686), (22,713), (8,724), (20,754), (23,777)\}.$

Παρατηρούμε ότι $(10,644) \in L_1$ *και* $(19,644) \in L_2$. *Επομένως* $\log_3 525 = 29 \cdot 10 + 19 = 309$.

Επαληθεύουμε και πράγματι $3^{309} = 525 (\mod 809)$.

4.2.2 Ο Pollard Rho αλγόριθμος

Ο αλγόριθμος Pollard Rho, είναι άλλος ένας αλγόριθμος που χρησιμοποιούμε για να λύσουμε το πρόβλημα του διακριτού λογαρίθμου. Έστω $(G, \cdot)$ μία ομάδα και $\alpha \in G$ ένα στοιχείο τάξεως n. Έστω επίσης $\beta \in < \alpha >$, το στοιχείο, του οποίου ψάχνουμε το διακριτό λογάριθμο. Αφού ο γεννήτορας $< \alpha >$ είναι κυκλικός, τάξεως n, τότε μπορούμε να θεωρήσουμε ότι το στοιχείο $\log_\alpha \beta$ είναι ένα στοιχείο του Z_n.

Σχηματίζουμε τώρα την ακολουθία $x_1, x_2, ...$, με την επαναληπτική εφαρμογή μιας τυχαίας συνάρτησης αναζήτησης, f. Μόλις βρούμε δύο στοιχεία της ακολουθίας, x_i και x_j, τέτοια ώστε $x_i = x_j$, με $i < j$, τότε μπορούμε να υπολογίσουμε το $\log_\alpha \beta$. Όπως και στον αλγόριθμο του Floyd, έτσι και εδώ αναζητούμε μία σύγκρουση της μορφής $x_i = x_{2i}$, ούτως ώστε να έχουμε τη λιγότερο δυνατή σπατάλη χρόνου και μνήμης.

Έστω $S_1 \cup S_2 \cup S_3$ τρία υποσύνολα περίπου ίδιου μεγέθους, να αποτελούν ένα διαμέρισμα της G. Ορίζουμε μία συνάρτηση $f : < \alpha > \times Z_n \times Z_n \rightarrow < \alpha > \times Z_n \times Z_n$ από τον παρακάτω τύπο :

$$f(x,a,b) = \left\{ \begin{array}{ll} (\beta \cdot x, a, b+1) & if x \in S_1 \\ (x^2, 2 \cdot a, 2 \cdot b) & if x \in S_2 \\ (\alpha \cdot x, a+1, b) & if x \in S_3 \end{array} \right\} \tag{4.7}$$

Επιπλέον κάθε τριπλέτα, (x, a, b), που δημιουργήσαμε, έχει την εξής ιδιότητα :

$$x = \alpha^a \cdot \beta^b \tag{4.8}$$

Έστω μία αρχική τριπλέτα που έχει αυτήν την ιδιότητα, η $(1, 0, 0)$. Παρατηρούμε ότι η $f(x, a, b)$ ικανοποιεί την (4.8), αν την ικανοποιεί και η τριπλέτα (x, a, b). Οπότε μπορούμε να ορίσουμε έναν γενικό τύπο ο οποίος έχει την εξής μορφή :

$$(x_i, a_i, b_i) = \left\{ \begin{array}{cc} (1,0,0) & if i = 0 \\ f(x_{i-1}, a_{i-1}, b_{i-1}) & if i \geq 1 \end{array} \right\} \tag{4.9}$$

Στη συνέχεια συγκρίνουμε τις τριπλέτες (x_{2i}, a_{2i}, b_{2i}) και (x_i, a_i, b_i), μέχρι να βρούμε ένα $i \geq 1$, τέτοιο ώστε :

$$x_{2i} = x_i \tag{4.10}$$

Όταν συμβεί η (4.10), από την (4.8) θα έχουμε :

$$\alpha^{a_{2i}} \cdot \beta^{b_{2i}} = \alpha^{a_i} \cdot \beta^{b_i} \tag{4.11}$$

Έστω $c = \log_\alpha \beta$. Επίσης γνωρίζουμε τις παρακάτω λογαριθμικές ιδιότητες :

1. $\alpha^{\log_\alpha(x)} = x$.
2. $\log_\alpha(x^y) = y \cdot \log_\alpha(x)$.

Από τα παραπάνω η (4.11) γίνεται :

$$\alpha^{a_{2i}+c\cdot b_{2i}} = \alpha^{a_i+c\cdot b_i}, \tag{4.12}$$

καθώς :

$$\begin{aligned} \alpha^{a_{2i}+c\cdot b_{2i}} &= \alpha^{a_{2i}+b_{2i}\cdot\log_\alpha(\beta)} \\ &= \alpha^{a_{2i}} \cdot \alpha^{\log_\alpha(\beta^{b_{2i}})} \\ &= \alpha^{a_{2i}} \cdot \beta^{b_{2i}} \end{aligned}$$

και

$$\begin{aligned} \alpha^{a_i+c\cdot b_i} &= \alpha^{a_i+b_i\cdot\log_\alpha(\beta)} \\ &= \alpha^{a_i+\log_\alpha(\beta^{b_i})} \\ &= \alpha^{a_i} \cdot \alpha^{\log_\alpha(\beta^{b_i})} \\ &= \alpha^{a_i} \cdot \beta^{b_i} \end{aligned}$$

Από την (4.12) προκύπτει η παρακάτω σχέση :

$$a_{2i} + c \cdot b_{2i} = a_i + c \cdot b_i \tag{4.13}$$

Γνωρίζουμε ότι το α έχει τάξη n οπότε η (4.13) γίνεται :

$$a_{2i} + c \cdot b_{2i} \equiv a_i + c \cdot b_i \pmod{n} \tag{4.14}$$

Η (4.14) μπορεί να γραφεί ως εξής :

$$c \cdot (b_{2i} - b_i) \equiv a_i - a_{2i} \pmod{n} \tag{4.15}$$

Εάν οι $b_{2i} - b_i, n$ είναι σχετικά πρώτοι μεταξύ τους, ισχύει δηλαδή Μ.Κ.Δ.$(b_{2i} - b_i, n) = 1$, τότε το $b_{2i} - b_i$ είναι αντιστρέψιμο στοιχείο στη Z_n. Οπότε μπορούμε να διαιρέσουμε την (4.15) με $b_{2i} - b_i$. Έτσι έχουμε :

$$c = (a_i - a_{2i}) \cdot (b_{2i} - b_i)^{-1} \pmod{n} \tag{4.16}$$

Είσοδος: Μία ομάδα $(G, \cdot)$, ένας γεννήτορας $\alpha \in G$, ένα $\beta \in < \alpha >$, η τάξη n του α, μία συνάρτηση $f :< \alpha > \times Z_n \times Z_n \rightarrow < \alpha > \times Z_n \times Z_n$, καθώς και τα διαστήματα S_1, S_2, S_3,
τέτοια ώστε : $G = S_1 \cup S_2 \cup S_3$.
Έξοδος: Ο λογάριθμος $\log_\alpha(\beta)$.

1. $i = 0$
2. $(x, a, b) = (1, 0, 0)$
3. **Επανέλαβε:**
4. $i = i + 1$
5. $(x_i, a_i, b_i) = f(x_{i-1}, a_{i-1}, b_{i-1})$
6. **Εάν :** $x \in S_1$
7. **τότε** $f = (\beta \cdot x, a, (b + 1) \pmod{n})$
8. **εάν** $x \in S_2$
9. **τότε** $f = (x^2, 2 \cdot a \pmod{n}, 2 \cdot b \pmod{n})$
10. **αλλιώς** $f = (\alpha \cdot x, (a + 1) \pmod{n}, b)$
11. **Μέχρις ότου:** $x_i = x_{2i}$
12. Τότε $\log_\alpha(\beta) = (a_i - a_{2i}) \cdot (b_{2i} - b_i)^{-1} \pmod{n}$

Σχήμα 4.3: Ο αλγόριθμος του Pollard Rho.

Παράδειγμα 4.2.2 *Ας δώσουμε τώρα ένα παράδειγμα του αλγορίθμου του Pollard Rho.*

Έστω $p = 809$ *ένας πρώτος αριθμός. Παρατηρούμε ότι το στοιχείο* $\alpha = 89$ *είναι τάξεως* $n = 101$ *στην* Z^*_{809} *και* $\beta = 618 \in < \alpha >$. *Θέλουμε να υπολογίσουμε το* $\log_{89}(618)$.

Αρχικά ορίζουμε τα σύνολα S_1, S_2, S_3 *ως εξής :*

$$S_1 = \{x \in Z^*_{809} : x \equiv 1 \pmod 3\}$$

$$S_2 = \{x \in Z^*_{809} : x \equiv 0 \pmod 3\}$$

$$S_3 = \{x \in Z^*_{809} : x \equiv 2 \pmod 3\}.$$

Στη συνέχεια υπολογίζουμε τις τριπλέτες (x_i, a_i, b_i) *και* (x_{2i}, a_{2i}, b_{2i}), *μέχρι να βρούμε* i, *τέτοιο ώστε* $x_i = x_{2i}$.

Για $i = 0$ *έχουμε από την (4.9),* $(x_0, a_0, b_0) = (1, 0, 0)$.

Για $i = 1$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_1, a_1, b_1) &= f(x_0, a_0, b_0) \\ &= f(1, 0, 0) \end{aligned} \tag{4.17}$$

Επίσης $1 \equiv 1 \pmod 3$, *οπότε* $x \in S_1$. *Οπότε η (4.17) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_1, a_1, b_1) &= (\beta \cdot 1, 0, 0 + 1) \\ &= (618, 0, 1)\end{aligned}$$

Για $i = 2$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_2, a_2, b_2) &= f(x_1, a_1, b_1) \\ &= f(618, 0, 1) \end{aligned} \tag{4.18}$$

Επίσης $618 \equiv 0 \pmod 3$, *οπότε* $x \in S_2$. *Οπότε η (4.18) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_2, a_2, b_2) &= (618^2, 2 \cdot 0, 2 \cdot 1) \\ &= (381924 (\mathrm{mod} 809), 0, 2) \\ &= (76, 0, 2)\end{aligned}$$

Παρατηρούμε όμως ότι $x_1 \neq x_2$, *άρα συνεχίζουμε την αναζήτηση του* i *και την παραγωγή τριπλετών.*

Για $i = 3$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_3, a_3, b_3) &= f(x_2, a_2, b_2) \\ &= f(76, 0, 2) \end{aligned} \tag{4.19}$$

Επίσης $76 \equiv 1 \pmod 3$, *οπότε* $x \in S_1$. *Οπότε η (4.19) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_3, a_3, b_3) &= (618 \cdot 76, 0, 2 + 1) \\ &= (46968(\text{mod}809), 0, 3) \\ &= (46, 0, 3)\end{aligned}$$

Για $i = 4$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_4, a_4, b_4) &= f(x_3, a_3, b_3) \\ &= f(46, 0, 3) \end{aligned} \tag{4.20}$$

Επίσης $46 \equiv 1 \pmod 3$, *οπότε* $x \in S_1$. *Οπότε η (4.20) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_4, a_4, b_4) &= (618 \cdot 46, 0, 3 + 1) \\ &= (28428(\text{mod}809), 0, 4) \\ &= (113, 0, 4)\end{aligned}$$

Παρατηρούμε όμως ότι $x_2 \neq x_4$, *άρα συνεχίζουμε.*

Για $i = 5$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_5, a_5, b_5) &= f(x_4, a_4, b_4) \\ &= f(113, 0, 4) \end{aligned} \tag{4.21}$$

Επίσης $113 \equiv 2 \pmod 3$, *οπότε* $x \in S_3$. *Οπότε η (4.21) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_5, a_5, b_5) &= (89 \cdot 113, 0 + 1, 4) \\ &= (10057(\text{mod}809), 1, 4) \\ &= (349, 1, 4)\end{aligned}$$

Για $i = 6$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_6, a_6, b_6) &= f(x_5, a_5, b_5) \\ &= f(349, 1, 4) \end{aligned} \tag{4.22}$$

Επίσης $349 \equiv 1 \pmod 3$, *οπότε* $x \in S_1$. *Οπότε η (4.22) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_6, a_6, b_6) &= (618 \cdot 349, 1, 4+1) \\ &= (215682(\mathrm{mod}809), 1, 5) \\ &= (488, 1, 5)\end{aligned}$$

Παρατηρούμε όμως ότι $x_3 \neq x_6$, *άρα συνεχίζουμε.*

Για $i = 7$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_7, a_7, b_7) &= f(x_6, a_6, b_6) \\ &= f(488, 1, 5)\end{aligned} \tag{4.23}$$

Επίσης $488 \equiv 2 \pmod 3$, *οπότε* $x \in S_3$. *Οπότε η (4.23) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_7, a_7, b_7) &= (89 \cdot 488, 1+1, 5) \\ &= (43432(\mathrm{mod}809), 2, 5) \\ &= (555, 2, 5)\end{aligned}$$

Για $i = 8$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_8, a_8, b_8) &= f(x_7, a_7, b_7) \\ &= f(555, 2, 5)\end{aligned} \tag{4.24}$$

Επίσης $555 \equiv 0 \pmod 3$, *οπότε* $x \in S_2$. *Οπότε η (4.24) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_8, a_8, b_8) &= (555^2, 2 \cdot 2, 2 \cdot 5) \\ &= (308025(\mathrm{mod}809), 4, 10) \\ &= (605, 4, 10)\end{aligned}$$

Παρατηρούμε όμως ότι $x_4 \neq x_8$, *άρα συνεχίζουμε.*

Για $i = 9$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_9, a_9, b_9) &= f(x_8, a_8, b_8) \\ &= f(605, 4, 10)\end{aligned} \tag{4.25}$$

Επίσης $605 \equiv 2 \pmod 3$, *οπότε* $x \in S_3$. *Οπότε η (4.25) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_9, a_9, b_9) &= (89 \cdot 605, 4+1, 10) \\ &= (53845(\mathrm{mod}809), 5, 10) \\ &= (451, 5, 10)\end{aligned}$$

Για $i = 10$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{10}, a_{10}, b_{10}) &= f(x_9, a_9, b_9) \\ &= f(451, 5, 10) \end{aligned} \tag{4.26}$$

Επίσης $451 \equiv 1 \pmod 3$, *οπότε* $x \in S_1$. *Οπότε η (4.26) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{10}, a_{10}, b_{10}) &= (618 \cdot 451, 5, 10 + 1) \\ &= (278718(\mathrm{mod}809), 5, 11) \\ &= (422, 5, 11)\end{aligned}$$

Παρατηρούμε όμως ότι $x_5 \neq x_{10}$, *άρα συνεχίζουμε.*

Για $i = 11$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{11}, a_{11}, b_{11}) &= f(x_{10}, a_{10}, b_{10}) \\ &= f(422, 5, 11) \end{aligned} \tag{4.27}$$

Επίσης $422 \equiv 2 \pmod 3$, *οπότε* $x \in S_3$. *Οπότε η (4.27) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{11}, a_{11}, b_{11}) &= (89 \cdot 422, 5 + 1, 11) \\ &= (37558(\mathrm{mod}809), 6, 11) \\ &= (344, 6, 11)\end{aligned}$$

Για $i = 12$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{12}, a_{12}, b_{12}) &= f(x_{11}, a_{11}, b_{11}) \\ &= f(344, 6, 11) \end{aligned} \tag{4.28}$$

Επίσης $344 \equiv 2 \pmod 3$, *οπότε* $x \in S_3$. *Οπότε η (4.28) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{12}, a_{12}, b_{12}) &= (89 \cdot 344, 6 + 1, 11) \\ &= (30616(\mathrm{mod}809), 7, 11) \\ &= (683, 7, 11)\end{aligned}$$

Παρατηρούμε όμως ότι $x_6 \neq x_{12}$, *άρα συνεχίζουμε.*

Για $i = 13$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{13}, a_{13}, b_{13}) &= f(x_{12}, a_{12}, b_{12}) \\ &= f(683, 7, 11) \end{aligned} \tag{4.29}$$

Επίσης $683 \equiv 2 \pmod 3$, *οπότε* $x \in S_3$. *Οπότε η (4.29) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{13}, a_{13}, b_{13}) &= (89 \cdot 683, 7+1, 11) \\ &= (60787(\text{mod}809), 8, 11) \\ &= (112, 8, 11)\end{aligned}$$

Για $i = 14$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{14}, a_{14}, b_{14}) &= f(x_{13}, a_{13}, b_{13}) \\ &= f(112, 8, 11) \qquad (4.30)\end{aligned}$$

Επίσης $112 \equiv 1 \pmod 3$, *οπότε* $x \in S_1$. *Οπότε η (4.30) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{14}, a_{14}, b_{14}) &= (618 \cdot 112, 8, 11+1) \\ &= (69216(\text{mod}809), 8, 12) \\ &= (451, 8, 12)\end{aligned}$$

Παρατηρούμε όμως ότι $x_7 \neq x_{14}$, *άρα συνεχίζουμε.*

Για $i = 15$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{15}, a_{15}, b_{15}) &= f(x_{14}, a_{14}, b_{14}) \\ &= f(451, 8, 12) \qquad (4.31)\end{aligned}$$

Επίσης $451 \equiv 1 \pmod 3$, *οπότε* $x \in S_1$. *Οπότε η (4.31) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{15}, a_{15}, b_{15}) &= (618 \cdot 451, 8, 12+1) \\ &= (278718(\text{mod}809), 8, 13) \\ &= (422, 8, 13)\end{aligned}$$

Για $i = 16$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{16}, a_{16}, b_{16}) &= f(x_{15}, a_{15}, b_{15}) \\ &= f(422, 8, 13) \qquad (4.32)\end{aligned}$$

Επίσης $422 \equiv 2 \pmod 3$, *οπότε* $x \in S_3$. *Οπότε η (4.32) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{16}, a_{16}, b_{16}) &= (89 \cdot 422, 8+1, 13) \\ &= (37558(\text{mod}809), 9, 13) \\ &= (344, 9, 13)\end{aligned}$$

Παρατηρούμε όμως ότι $x_8 \neq x_{16}$, *άρα συνεχίζουμε.*

Για $i = 17$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{17}, a_{17}, b_{17}) &= f(x_{16}, a_{16}, b_{16}) \\ &= f(344, 9, 13) \end{aligned} \tag{4.33}$$

Επίσης $344 \equiv 2 \pmod 3$, *οπότε* $x \in S_3$. *Οπότε η (4.33) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{17}, a_{17}, b_{17}) &= (89 \cdot 344, 9 + 1, 13) \\ &= (30616(\text{mod}809), 10, 13) \\ &= (683, 10, 13)\end{aligned}$$

Για $i = 18$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{18}, a_{18}, b_{18}) &= f(x_{17}, a_{17}, b_{17}) \\ &= f(683, 10, 13) \end{aligned} \tag{4.34}$$

Επίσης $683 \equiv 2 \pmod 3$, *οπότε* $x \in S_3$. *Οπότε η (4.34) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{18}, a_{18}, b_{18}) &= (89 \cdot 683, 10 + 1, 13) \\ &= (60787(\text{mod}809), 11, 13) \\ &= (112, 11, 13)\end{aligned}$$

Παρατηρούμε όμως ότι $x_9 \neq x_{18}$, *άρα συνεχίζουμε.*

Για $i = 19$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{19}, a_{19}, b_{19}) &= f(x_{18}, a_{18}, b_{18}) \\ &= f(112, 11, 13) \end{aligned} \tag{4.35}$$

Επίσης $112 \equiv 1 \pmod 3$, *οπότε* $x \in S_1$. *Οπότε η (4.35) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{19}, a_{19}, b_{19}) &= (618 \cdot 112, 11, 13 + 1) \\ &= (69216(\text{mod}809), 11, 14) \\ &= (451, 11, 14)\end{aligned}$$

Για $i = 20$ *και με τη χρήση της (4.9), έχουμε :*

$$\begin{aligned}(x_{20}, a_{20}, b_{20}) &= f(x_{19}, a_{19}, b_{19}) \\ &= f(451, 11, 14) \end{aligned} \tag{4.36}$$

Επίσης $451 \equiv 1 \pmod 3$, *οπότε* $x \in S_1$. *Οπότε η (4.36) μέσω της (4.7) γίνεται :*

$$\begin{aligned}(x_{20}, a_{20}, b_{20}) &= (618 \cdot 451, 11, 14 + 1) \\ &= (278718(\text{mod}809), 11, 15) \\ &= (422, 11, 15)\end{aligned}$$

Παρατηρούμε ότι $x_{10} = x_{20}$*, οπότε σταματάμε και από την (4.16) έχουμε :*

$$\begin{aligned} c &= (5-11)\cdot(15-11)^{-1}(\mathrm{mod}101) \\ &= (-6\cdot 4^{-1})(\mathrm{mod}101) \\ &= 49 \end{aligned}$$

Άρα $\log_{\alpha}(\beta) = 49$ *στην* Z^{*}_{809}.

Βιβλιογραφία

[1] Richard P. Brent, An Improved Monte Carlo Factorization Algorithm. *BIT 20*, pages 176–184, 1980.

[2] Γκρίτζαλης Στέφανος, Κάτσικας Σωκράτης και Χρυσικόπουλος Βασίλειος, Burmester M, Σύγχρονη κρυπτογραφία Θεωρία και εφαρμογές, *Παπασωτηρίου*, 2011.

[3] Michael R. Garey and David S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, *Macmillan Higher Education*, 1979.

[4] Michael Sipser, Introduction to the Theory of Computation, Second Edition, *PWS*, 2006.

[5] Douglas R. Stinson, Cryptography Theory and Practice, Third Edition, *Chapman and Hall/CRC* , 2005.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Introduction to Algorithms, *MIT Press*, 2009.

[7] J. M. Pollard, A Monte Carlo Method for Factorization, *BIT 15*, pages 331–334, 1975.

[8] Robert Sedgewick, Thomas G. Szymanski, The Complexity of Finding Periods