Νίκη Βάζου, Ελένη Μπακάλη

21 Ιουλίου 2011

Κρυπτογραφία :

Θεωρία Ομάδων στην Κρυπτογραφία

Μεταπτυχιακή Εργασία για το μάθημα Κρυπτογραφία και Μαθηματικά.
Τμήμα Μαθηματικών Ε.Κ.Π.Α.
Διδάσκων : Ευάγγελος Ράπτης

1 Εισαγωγή

1.1 Κρυπτοσυστήματα

Ας υποθέσουμε ότι δύο άτομα απομακρυσμένα μεταξύ τους θέλουν να επικοινωνήσουν με τέτοιο τρόπο που κανείς άλλος να μην μάθει τι είπανε. Επειδή είναι απομακρυσμένοι, το μήνυμα αναγκαστικά θα ταξιδέψει κι έτσι θα είναι εκτεθειμένο σε οποιονδήποτε. Μια λύση για να το προστατέψουν είναι να το διαμορφώσουν έτσι ώστε ακόμα και αν κάποιος το δει, να μην μπορεί να καταλάβει τι λέει. Αυτό το σκοπό εξυπηρετούν τα διάφορα κρυπτοσυστήματα.

Ένα κρυπτοσύστημα είναι μια συνάρτηση $f_E : \mathcal{P} \to \mathcal{C}$ από το σύνολο των απλών μηνυμάτων $\mathcal{P}$ στο σύνολο $\mathcal{C}$ των κρυπτογραφημάτων. Αυτή συνήθως εξαρτάται από κάποιες παραμέτρους τις οποίες ονομάζουμε <κλειδί> του κρυπτοσυστήματος. Η συνάρτηση f_E είναι δημοσίως γνωστή, ενώ το κλειδί Ε κρατείται μυστικό. Επίσης η αντίστροφη $f_E^{-1} \equiv f_D$ καλείται συνάρτηση αποκρυπτογράφησης, με αντίστοιχο κλειδί D και είναι επίσης δημοσίως γνωστή.

Ένα παράδειγμα είναι το κρυπτοσύστημα του Καίσαρα σύμφωνα με το οποίο κρυπτογραφούμε κάθε μήνυμα χωριστά ως εξής: το A γίνεται D, το B γίνεται E, το C γίνεται F κ.ο.κ. Αν αντιστοιχίσουμε στο Α το 1, στο Β το 2 κλπ., τότε η συνάρτηση κρυπτογράφησης είναι η $f(x) = x + 3 \ mod \ 26$ και η συνάρτηση αποκρυπτογράφησης είναι η $f^{-1}(x) = x - 3 \ mod \ 26$. Γενικότερα θα μπορούσαμε να έχουμε το κρυπτοσύστημα $f_b(x) = x + b \, mod \, 26, f_b^{-1}(x) = x + (-b) \ mod \, 26$, με κλειδί κρυπτογράφησης και αποκρυπτογράφησης το b.

Σε αυτό το κρυπτοσύστημα, όπως συνέβαινε και σε όλα μέχρι τη δεκαετία του '70, όπως το DES και το AES, το κλειδί για την αποκρυπτογράφηση ήταν το ίδιο με της κρυπτογράφησης, ή μπορούσε κάποιος να υπολογίσει εύκολα το ένα από το άλλο. Τέτοια κρυπτοσυστήματα ονομάζονται συμμετρικά. Για να επιτευχθεί η επικοινωνία, πρέπει οι δύο πλευρές με κάποιον τρόπο να συμφωνήσουν σε ένα κοινό κλειδί. Από την άλλη πλευρά, αν κάποιος τρίτος καταφέρει να αποκτήσει το κλειδί κρυπτογράφησης, μπορεί εύκολα να υπολογίσει και το κλειδί αποκρυπτογράφησης (και το αντίστροφο).

1.2 Κρυπτογραφία δημοσίου κλειδιού

Η ιδέα πίσω από την κρυπτογραφία δημοσίου κλειδιού, την οποία πρώτοι εμπνεύστηκαν οι Diffie και Hellman το 1976, ήταν το να μην είναι εύκολο να υπολογίσει κανείς το κλειδί αποκρυπτογράφησης D γνωρίζοντας το κλειδί κρυπτογράφησης E (δηλαδή να μην υπάρχει αλγόριθμος πολυωνυμικού χρόνου που να το υπολογίζει). Ή αλλιώς, να είναι υπολογιστικά δύσκολο να αντιστρέψει κανείς την $f_E(x)$, εκτός αν κατέχει κάποια πρόσθετη πληροφορία, όπως το D.

Οπότε το E μπορεί να είναι δημοσίως γνωστό και μόνο το D να κρατείται κρυφό. Έτσι δεν χρειάζεται να προηγηθεί συμφωνία σε κάποιο κοινό κλειδί: Κάθε πλευρά (i) επιλέγει το ιδιωτικό (κρυφό) κλειδί d_i, και υπολογίζει και δημοσιοποιεί το αντίστοιχο δημόσιο e_i. Φυσικά αυτός ο υπολογισμός θα πρέπει να είναι εφικτός, αλλά όπως είπαμε ήδη, η εύρεση του d_i από το e_i πρέπει να είναι υπολογιστικά δύσκολη.

Για να στείλει ο Bob (Β) ένα μήνυμα στην Alice (Α), κρυπτογραφεί με το e_A. Σύμφωνα με

τα παραπάνω, κανείς δεν θα μπορεί να διαβάσει το μήνυμα του Β, εκτός από την Α, η οποία είναι η μόνη που γνωρίζει το κλειδί αποκρυπτογράφησης d_A. Φυσικά ο Β δεν χρειάζεται να γνωρίζει το d_A. Τώρα αν η Α θέλει να απαντήσει στον Β, θα κρυπτογραφήσει με το e_B και μόνο ο Β θα μπορεί να διαβάσει το μήνυμα, χρησιμοποιώντας το d_B.

Παρατήρηση 1 Στην πράξη τα κρυπτοσυστήματα δημοσίου κλειδιού χρησιμοποιούν κλειδιά πολύ μεγαλύτερου μήκους από αυτά της συμμετρικής κρυπτογραφίας. Επιπλέον η κρυπτογράφηση στην πρώτη περίπτωση είναι πιο χρονοβόρα διαδικασία από ότι στην δεύτερη, ειδικά όταν έχουμε μηνύματα μεγάλου μήκους.

Για αυτόν το λόγο η κρυπτογραφία δημοσίου κλειδιού συνήθως χρησιμοποιείται για να επιτευχθεί η συμφωνία σε ένα κοινό κλειδί, το οποίο στη συνέχεια θα χρησιμοποιηθεί σε ένα σύστημα συμμετρικής κρυπτογραφίας.

Για παράδειγμα ένα πρωτόκολλο ανταλλαγής κλειδιού που βασίζεται σε κρυπτογραφία δημοσίου κλειδιού είναι το πρωτόκολλο Diffie Hellman, που βασίζεται στη δυσκολία του διακριτού λογαρίθμου, δηλ. της αντιστροφής της $f(x) = g^x \mod p$ όπου p πρώτος:

Έστω Z_p και $g \in Z_p$ ένα δημόσια γνωστό στοιχείο. Αν η Αλίκη και ο Βασίλης θέλουν να συμφωνήσουν ένα κοινό κρυφό κλειδί, μπορούν να κάνουν τα εξής:

1. Η Αλίκη διαλέγει τυχαία και μυστικά ένα στοιχείο $\alpha \in \{2, \ldots, p-1\}$, υπολογίζει το $g^\alpha \mod p$ και το στέλνει στον Βασίλη.
2. Ο Βασίλης διαλέγει τυχαία και μυστικά ένα στοιχείο $\beta \in \{2, \ldots, p-1\}$, υπολογίζει το $g^\beta \mod p$ και το στέλνει στην Αλίκη.
3. Η Αλίκη υπολογίζει το $k_\alpha = \left(g^\beta\right)^\alpha \mod p$.
4. Ο Βασίλης υπολογίζει το $k_\beta = (g^\alpha)^\beta \mod p$.
5. Το κοινό κλειδί είναι το $k = k_\alpha = k_\beta$.

Παρατήρηση 2 Όπως αναφέραμε, τα κρυπτοσυστήματα δημοσίου κλειδιού βασίζουν την ασφάλειά τους στη δυσκολία αντιστροφής μιας συνάρτησης f, εκτός αν υπάρχει γνώση μιας επιπλέον πληροφορίας. Τέτοιες συναρτήσεις λέγονται συναρτήσεις καταπακτής (trapdoor functions). Παρεμφερής είναι και η έννοια της συνάρτησης μονής κατεύθυνσης (one way function). Έτσι ονομάζεται μια συνάρτηση, με την ιδιότητα ότι είναι εύκολο να βρεις την τιμή της σε ένα σημείο, αλλά είναι δύσκολο να την αντιστρέψεις.

Είναι ανοιχτό πρόβλημα αν υπάρχουν τέτοιες συναρτήσεις. Αυτό συνδέεται με προβλήματα της θεωρίας πολυπλοκότητας, τα οποία είναι ακόμα ανοιχτά μετά από δεκαετίες προσπαθειών να απαντηθούν. Ένα τέτοιο πρόβλημα είναι η εικασία $P \neq NP$. Τελικά αυτά τα κρυπτοσυστήματα βασίζουν την ασφάλειά τους στο γεγονός ότι *μέχρι τώρα* δεν έχει βρεθεί πολυωνυμικού χρόνου αλγόριθμος που να αντιστρέφει τις συναρτήσεις που χρησιμοποιούν.

Τα τελευταία χρόνια έχουν προταθεί πολλά κρυπτοσυστήματα που βασίζονται σε προβλήματα της θεωρίας ομάδων.

2 Κρυπτογραφία με χρήση Ομάδων

Θα περιγράψουμε κάποια σχήματα που έχουν προταθεί, τα οποία χρησιμοποιούν μη αβελιανές ομάδες (μη αντιμεταθετικές), όπως αυτά παρουσιάζονται στο [BCM09]. Το δύσκολο πρόβλημα στο οποίο βασίζουν την ασφάλειά τους είναι το πρόβλημα της συζυγίας.

2.1 Συζυγία και Ύψωση σε Δύναμη

Έστω G μια μη αβελιανή ομάδα. Για $g, x \in G$ ορίζουμε

$$g^x = x^{-1}gx$$

τον συζυγή του g με τον x. Αυτή η γραφή υπονοεί ότι η συζυγία μπορεί να χρησιμοποιηθεί αντί της ύψωσης σε δύναμη στην κρυπτογραφία. Έτσι, μπορούμε να ορίσουμε το ανάλογο πρόβλημα του διακριτού λογαρίθμου:

Πρόβλημα Αναζήτησης συζηγους: Έστω G μία μη αβελιανή ομάδα. Έστω $g, h \in G$ τέτοια ώστε $h = g^x$ για κάποιο άγνωστο $x \in G$. Δοσμένων των g και h, βρες ένα στοιχείο $y \in G$ τέτοιο ώστε $h = g^y$. Ή με τον άλλο συμβολισμό, δεδομένων των g, h βρες y τ.ω. $h = y^{-1}gy$.

Αν βρούμε κάποια ομάδα στην οποία το Πρόβλημα Αναζήτησης Συζυγούς είναι υπολογιστικά δύσκολο (και τα στοιχεία της ομάδας είναι εύκολο να τα αποθηκεύσουμε και να τα διαχειριστούμε) τότε μπορούμε να ορίσουμε κρυπτοσυστήματα ανάλογα αυτών που στηρίζονται στο πρόβλημα του διακριτού λογαρίθμου. Οι Ko et al. πρότειναν το ακόλουθο ανάλογο του πρωτοκόλλου ανταλλαγής κλειδιού Diffie-Hellman.

Πρωτόκολλο ανταλλαγής κλειδιού Ko-Lee-Cheon-Han-Kang-Park [KLC+00] Έστω G μια μη αβελιανή ομάδα και $g \in G$ ένα δημόσια γνωστό στοιχείο της G. Έστω A, B δύο commuting υποομάδες της G, δηλ. τα a, b αντιμετατίθενται για κάθε $a \in A, b \in B$. Αν η Αλίκη και ο Βασίλης θέλουν να συμφωνήσουν ένα κοινό κρυφό κλειδί, μπορούν να κάνουν τα εξής:

1. Η Αλίκη διαλέγει τυχαία και μυστικά ένα στοιχείο $\alpha \in A$, υπολογίζει το $g^\alpha = \alpha^{-1}g\alpha$ και το στέλνει στον Βασίλη.
2. Ο Βασίλης διαλέγει τυχαία και μυστικά ένα στοιχείο $\beta \in B$, υπολογίζει το $g^\beta = \beta^{-1}g\beta$ και το στέλνει στην Αλίκη.
3. Η Αλίκη υπολογίζει το $k_\alpha = \left(g^\beta\right)^\alpha$.
4. Ο Βασίλης υπολογίζει το $k_\beta = (g^\alpha)^\beta$.
5. Το κοινό κλειδί είναι το $k = k_\alpha = k_\beta$, αφού $\alpha\beta = \beta\alpha$.

Πρέπει να παρατηρήσουμε ότι τα k_α, k_β μπορεί να έχουν διαφορετική αναπαράσταση, και γενικά δεν είναι εύκολο να ελέγξει κανείς αν δύο στοιχεία με διαφορετική αναπαράσταση ταυτίζονται. Οπότε χρειαζόμαστε ομάδες των οποίων τα στοιχεία να μπορούν να

γραφτούν σε μία (μοναδική) κανονική μορφή. Η ομάδες που έχουν προταθεί για αυτό το πρωτόκολλο είναι οι ομάδες πλεξίδων (braid groups), τις οποίες θα παρουσιάσουμε αργότερα.

Μία παραλλαγή του πρωτοκόλλου είναι το ακόλουθο των Anshel, Anshel και Goldfeld που έχει το πλεονέκτημα ότι οι commuting υποομάδες A και B δεν χρειάζονται.

Πρωτόκολλο ανταλλαγής κλειδιού Anshel-Anshel-Goldfeld [AAG99] Έστω G μια μη αβελιανή ομάδα και έστω τα στοιχεία $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_m \in G$ δημόσια γνωστά στοιχεία της G. Αν η Αλίκη και ο Βασίλης θέλουν να συμφωνήσουν ένα κοινό κρυφό κλειδί, μπορούν να κάνουν τα εξής:

1. Η Αλίκη διαλέγει τυχαία και μυστικά ένα στοιχείο $x \in \alpha_1, \ldots, \alpha_k$, υπολογίζει τα $\beta_1^x, \ldots, \beta_m^x$ και τα στέλνει στον Βασίλη.
2. Ο Βασίλης διαλέγει τυχαία και μυστικά ένα στοιχείο $y \in \beta_1, \ldots, \beta_m$, υπολογίζει τα $\alpha_1^y, \ldots, \alpha_k^y$ και τα στέλνει στην Αλίκη.
3. Η Αλίκη υπολογίζει το $k_\alpha = x^y$.
4. Ο Βασίλης υπολογίζει το $k_\beta = y^x$.
5. Το κοινό κλειδί είναι το $k = [x, y] = x^{-1}y^{-1}xy = x^{-1}k_\alpha = \left(y^{-1}k_\beta\right)^{-1}$

Από το βήμα 5 φάινεται ότι δεν είναι απαραίτητο να λύσει η Alice (ή ο Bob αντίστοιχα) το πρόβλημα της συζυγίας, για να μπορέσει να βρει το y (το x ο Bob αντίστοιχα) και να υπολογίσει το κλειδί.

Για αυτό το πρωτόκολλο προτάθηκε και πάλι η χρήση των Braid groups, λόγω (α) της ύπαρξης κανονικής μορφής, και (β) της δυσκολίας του προβλήματος της συζυγίας σε αυτές τις ομάδες. Παρακάτω θα δώσουμε συνοπτικά και άλλες πιθανές ομάδες που έχουν προταθεί να χρησιμοποιηθούν σε αυτά τα πρωτόκολλα.

2.2 Platform groups

Για να είναι ασφαλές κάποιο "κανονικό" κρυπτογραφικό πρωτόκολλο (που στηρίζεται σε one-way συνάρτηση), πρέπει η group platform πάνω στην οποία υλοποιείται να ικανοποιεί κάποιες απαιτήσεις. Για το 2.1 οι απαιτήσεις αυτές μπορούν να διατυπωθούν ως εξής:

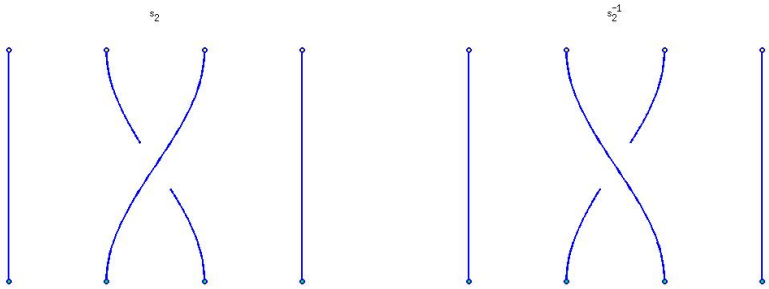
1. Η ομάδα πρέπει να είναι γνωστή ή/και να έχει μελετηθεί αρκετά.
2. Ο υπολογισμός του κλειδιού πρέπει να είναι υπολογιστικά εύκολος.
3. Το πρόβλημα αναζήτησης συζυγούς πρέπει να είναι υπολογιστικά δύσκολο.
4. Το πλήθος των λέξεων μήκους n της ομάδας, πρέπει να είναι υπερεκθετικό του n.

Η ύπαρξη των τριών πρώτων απαιτήσεων είναι προφανής. Η τέταρτη απαίτηση εισάγεται για να αποκλειστεί το ενδεχόμενο το πρόβλημα Αναζήτησης συζυγούς να λυθεί αποδοτικά με την εξαντλητική (brute-force) μέθοδο.

Στην συνέχεια της ενότητας θα παρουσιάσουμε έξι ομάδες που ικανοποιούν τις απαιτήσεις αυτές, άρα μπορούν να χρησιμοποιηθούν για την υλοποίηση των πρωτοκόλλων 2.1. Θα δώσουμε έμφαση στις Braid groups, ενώ τις υπόλοιπες θα αναφέρουμε επιγραμματικά.

2.2.1 Braid Groups

Η braid group στο n συμβολίζεται με B_n, είναι μία άπειρη ομάδα για $n > 1$. Τα braid groups έχουν μελετηθεί εκτενώς [GAR, EM94, EPC+92] και έχουν την ακόλουθη γεωμετρική ερμηνεία. Θεωρούμε δύο ραβδιά με n καρφιά η καθεμία, το ένα απέναντι από το άλλο, έτσι ώστε κάθε καρφί του ενός ραβδιού να βρίσκεται απέναντι από κάποιο καρφί του άλλου. Χρησιμοποιώντας νήματα, συνδέουμε κάθε καρφί του ενός ραβδιού με κάποιο του άλλου δημιουργώντας μία ένα-προς-ένα αντιστοιχία. Συχνά κάποιο νήμα θα πρέπει να περάσει κάτω ή πάνω από κάποιο άλλο, μία ή περισσότερες φορές, και αυτό είναι σημαντικό, αφού οι ακόλουθες πλεξίδες είναι διαφορετικές:



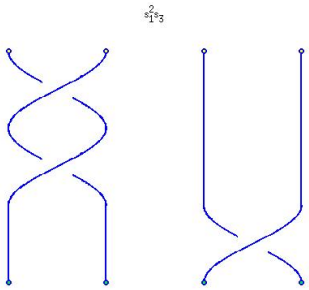
Σχήμα 1: Braids σ_2 και σ_2^{-1}

Οι γεννήτορες της ομάδας συμβολίζονται με $\sigma_j, 0 < j < n$, και σ_j είναι η πλεξίδα που ενώνει κάθε καρφί με το απέναντί του, εκτός από το j-οστό που το ενώνει με το $j+1$, ενώ το $j+1$ με το j, και επιπλέον το j-οστό νήμα τοποθετείται κάτω από το $j+1$. Αντίστροφό του είναι το σ_j^{-1}, όπου τα δύο νήματα διασταυρώνονται αντίθετα, όπως στο Σχήμα 1, που βλέπουμε τα $\sigma_2, \sigma_{-2}, (n = 4)$.

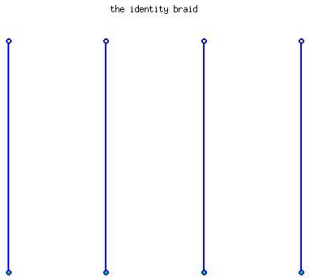
Αν ενώσουμε δύο braids u και v στην σειρά τότε παίρνουμε ένα καινούργιο, το uv, άρα η συνένωση συμβολίζει τον πολλαπλασιασμό. Για παράδειγμα, στο Σχήμα 2 βλέπουμε την $\sigma_1\sigma_1\sigma_3$.

Σαν ουδέτερο στοιχείο της πράξης θεωρούμε αυτό που δεν περιέχει καθόλου τομές και το ονομάζουμε απλό braid, όπως στο Σχήμα 3.

Θεωρούμε δύο braids ισοδύναμα αν είναι δυνατό να δημιουργήσουμε από το ένα το άλλο μετακινώντας τα νήματα στον χώρο χωρίς να αλλάζουμε τα άκρα τους. Το σύνολο B_n είναι το σύνολο των κλάσεων ισοδυναμίας των braids με n νήματα και έχει δομή ομάδας αφού αν πολλαπλασιάσουμε κάποιο braid με την αντανάκλασή (τον αντίστροφό) του



Σχήμα 2: Braid $\sigma_1\sigma_1\sigma_3$



Σχήμα 3: Απλό braid

το αποτέλεσμα είναι ισοδύναμο με το απλό braid.

Ουσιαστικά κάθε braid είναι μία ακολουθία τομών. Ονομάζουμε μια τομή θετική αν το μπροστινό νήμα έχει θετική κλίση (όπως η σ_2), διαφορετικά την ονομάζουμε αρνητική (όπως η σ_{-2}).

Εύκολα προκύπτει ότι ισχύουν τα εξής:

$$[\sigma_i, \sigma_j] = 1,$$

για κάθε i, j τέτοια ώστε $\mid i - j \mid > 1$ και

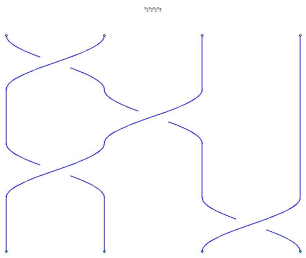
$$x_i x_{i+1} x_i = x_{i+1} x_i x_{i+1}$$

για κάθε i τέτοιο ώστε $1 \leq i \leq n-2$

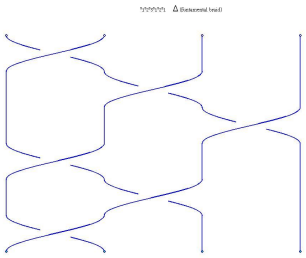
Στην πραγματικότητα οι δύο αυτές σχέσεις αρκούν για να περιγράψουν την B_n.

Permutation braid έχουμε όταν όλες οι τομές της πλεξίδας είναι θετικές, και κάθε δύο νήματα διασταυρώνονται το πολύ μία φορά, όπως στο Σχήμα 4 που φαίνεται η $\sigma_1\sigma_2\sigma_1\sigma_3$ που αντιστοιχεί στην permutation 4213. Η permutation braid που αντιστοιχεί στην μετάθεση 4321 (γενικά στην μετάθεση n,n-1,...,2,1), λέγεται fundamental braid και θα την συμβολίζουμε με Δ και φαίνεται στο Σχήμα 5.

Το σύνολο των permutation braids, θα το συμβολίζουμε με Σ_n.



Σχήμα 4: Braid $\sigma_1\sigma_2\sigma_1\sigma_3$



Σχήμα 5: Briad Δ

Για την B_n υπάρχουν αρκετές γνωστές κανονικές μορφές, οι πιο γνωστές εκ των οποίων είναι οι Dehornoy handle free form και η Garside normal form. Για την Dehornoy handle free form δεν υπάρχει καλή θεωρητική εκτίμηση πολυπλοκότητας, όμως στην πράξη δουλεύει σε γραμμικό χρόνο, ενώ η Garside normal form τρέχει σε τετραγωνικό χρόνο. Η ύπαρξη πολλών κανονικών μορφών μπορεί να προκαλέσει πρόβλημα, αφού κάποια κανονική μορφή μπορεί να ;αποκαλύπτει πληροφορίες που κάποια άλλη προσπαθεί να αποκρύψει. Συγκεκριμένα, έχει αποδειχτεί ότι η μετατροπή από την Garside στην Dehornoy κανονική μορφή κάνει τις υποκείμενες μεταδόσεις πιο επιρρεπείς σε επιθέσεις.

Η δυσκολία του προβλήματος της συζυγίας φαίνεται να ικανοποιείται, αν και παρά τις μελέτες που έχουν γίνει είναι ακόμα άγνωστο αν το πρόβλημα αναζήτησης συζηγούς ανήκει στο NP. Τελικά, όσον αφορά την ιδιότητα 4 το πλήθος των $B_n, n \geq 2$ έχουν εκθετική αύξηση. Για $n \geq 3$ αυτό προκύπτει από το γεγονός ότι η B_n έχει ελεύθερες υποομάδες, για παράδειγμα οι σ_1^2 και σ_2^2 παράγουν μια ελεύθερη υποομάδα.

Παρατηρήσεις

1. Έστω B_n^+ το σύνολο των θετικών πλεξίδων (δηλ. των πλεξίδων που όλες οι τομές είναι θετικές). Για μία θετική πλεξίδα P ορίζουμε το starting set, και το finishing set

$$S(P) = \{i | P = \sigma_i P' \; for \; some \; P' \in B_n^+\}$$

$$F(P) = \{i | P = P'\sigma_i \; for \; some \; P' \in B_n^+\}.$$

Για μία permutation braid A που αντιστοιχεί στην permutation π (όπου $\pi(i) = j$ αν το i καρφί του πάνω ραβδιού ενώνεται με το j του κάτω) ισχύει

$$S(A) = \{i|\pi(i) > \pi(i+1)\}$$

$$F(A) = \{i|\pi^{-1}(i) > \pi^{-1}(i+1)\}.$$

Για παράδειγμα, για την Δ στην φωτογραφία $S(\Delta) = F(\Delta) = \{1, 2, 3\}$, ενώ για την $A = \sigma_1\sigma_2\sigma_1\sigma_3$ είναι $S(A) = \{1, 2\}, F(A) = \{1, 3\}$.

2. Για την Δ πλεξίδα ισχύουν τα εξής

   (i) $\forall 0 < j < n, \Delta = \sigma_j A_j = B_j\sigma_j$ για κάποιες permutation braids A_j, B_j.

   (ii) $\forall 0 < i < n, \sigma_i\Delta = \Delta\sigma_i$.

   Άρα σε κάθε λέξη (πλεξίδα) W από σ_i, μπορούμε να αντικαταστήσουμε τα σ_i^{-1} με τον τύπο $\sigma_i^{-1} = \Delta^{-1}B_i$ από την πρώτη ιδιότητα, και χρησιμοποιώντας την δεύτερη ιδιότητα, να φέρουμε τα Δ στην αρχή. Έτσι παίρνουμε την έκφραση $W = \Delta^u P, P \in B_n^+$.

3. Για κάθε θετική λέξη P υπάρχει μία μοναδική αποσύνθεση που λέγεται left-weighted decomposition, ως εξής

   $$P = A_1P_1, A_1 \in \Sigma_n, P_1 \in B_n^+, S(P_1) \subset F(A_1).$$

   Επαναλαμβάνοντας την ίδια διαδικασία για το P_1 κ.ο.κ. και φέρνοντας τα Δ στην αρχή, παίρνουμε την αριστερή κανονική μορφή Garside, η οποία είναι μοναδική για κάθε πλεξίδα

   $$P = \Delta^u A_1A_2...A_p, u \in Z, A_i \in \Sigma_n \setminus \{e, \Delta\}.$$

Από τις παραπάνω παρατηρήσεις προκύπτει το εξής θεώρημα κανονικής μορφής.

Theorem 2.2.1. *Για κάθε $W \in B_n$ υπάρχει μία μοναδική αναπαράσταση, που λέγεται αριστερή κανονική μορφή*

$$P = \Delta^u A_1A_2...A_p, u \in Z, A_i \in \Sigma_n \setminus \{e, \Delta\},$$

όπου A_iA_{i+1} είναι left-weighted για κάθε $0 < i < n$. Το p ορίζεται ως το μήκος Garside $l(W)$.

Παρατήρηση Αποδεικνύεται ότι η παραπάνω κανονική μορφή μπορεί να υπολογιστεί σε χρόνο $O(l^2n\log n)$, όπου l το μήκος της λέξης σε μη κανονική μορφή. Επίσης αποδεικνύεται ότι το πλήθος των n-braids κανονικού μήκους p είναι τουλάχιστον $(\lfloor\frac{n-1}{2}\rfloor!)^p$.

2.2.2 Thompson's group

Η ομάδα F του Thomson [Bel07] είναι μία άπειρη μη αβελιανή ομάδα που είναι γνωστή σε πολλούς κλάδους των μαθηματικών, όπως η άλγεβρα, η γεωμετρία και η ανάλυση. Για τον λόγω αυτό, η ομάδα έχει μελετηθεί εκτενώς και μπορεί να προταθεί για την υλοποίηση των 2.1.

Η ομάδα αυτή μπορεί να περιγραφεί με χρήση γεννητόρων και σχέσεων ορισμού ως

$$F = \langle x_0, x_1, x_2 \mid x_i^{-1} x_k x_i = x_{k+1}\ (k > i) \rangle$$

Όπως και στις braid ομάδες, στην ομάδα F υπάρχουν πολλές κανονικές μορφές, όμως η "κλασική" κανονική μορφή ενός στοιχείου w μπορεί να υπολογιστεί σε χρόνο $O(\mid w \mid \log \mid w \mid)$, άρα ικανοποιείται και η 2 ιδιότητα.

Όσον αφορά την 3η ιδιότητα αν και στο [MK06] αποδεικνύεται ότι υπάρχει αποδοτική λύση για την επίλυση του ταυτόχρονου προβλήματος του συζυγούς στο F, το πρόβλημα ιδιότητας μέλους παραμένει απρόσιτο, συνεπώς δεν υπάρχει αλγόριθμος που να επιτρέπει σε κάποιον αντίπαλο αποδοτικά να υπολογίσει το κλειδί.

Όσον αφορά την 4η ιδιότητα μπορεί να δειχθεί [JWCP96] ότι ικανοποιείται.

2.2.3 Ομάδες Πινάκων

Σαν πλατφόρμα για την υλοποίηση των 2.1 μπορούν να χρησιμοποιηθούν ομάδες πινάκων πάνω σε πεπερασμένους αντιμεταθετικούς δακτύλιους αφού αυτές έχουν ότι συνδυάζουν τα πλεονεκτήματα και τα μειονεκτήματα της αντιμεταθετικότητα. Συγκεκριμένα, ο πολλαπλασιασμός πινάκων είναι μη αντιμεταθετικός, αλλά τα στοιχεία των πινάκων προέρχονται από αντιμεταθετικό δακτύλιο και συνεπώς μπορούν να παρέχουν μία "φυσική" κανονική μορφή.

Ο συνδυασμός αντιμεταθετικότητας των στοιχείων των πινάκων και μη-αντιμεταθετικότητας του πολλαπλασιασμού είναι πολύ σημαντικό συστατικό της ασφάλειας του συστήματος, αφού απαγορεύει στον εισβολέα να χρησιμοποιήσει προφανείς σχέσεις όπως η $ab = ba$ για να απλοποιήσει γινόμενα.

2.2.4 Άλλες Ομάδες

Άλλες ομάδες που μπορούν να χρησιμοποιηθούν είναι οι small cancellation groups, solvable groups, Artin groups, Grogorchuck's group.

3 Κρυπτανάλυση

Σε αυτήν την ενότητα θα αναφέρουμε κάποιες επιθέσεις στο πρόβλημα της συζυγίας στις Braid groups. Το 1969 ο Garside [Gar69] έδωσε έναν αλγόριθμο για την εύρεση αν δύο στοιχεία της ομάδας είναι συζυγή, κατασκευάζοντας για κάθε στοιχείο x ένα υποσύνολο I_x (που ονομάζεται summit set) τ.ω. τα x, y είναι συζυγή αν και μόνο αν $I_x = I_y$. Το πρόβλημα είναι ότι τα summit sets μπορεί να ήταν εκθετικά μεγάλα.

Άλλη επίθεση (των Hofheinz και Steinwandt [HS02]) βασίζεται στην παρατήρηση ότι οι αντιπρόσωποι δύο συζυγών μέσα σε μία ειδική κατηγορία υποσυνόλων που λέγονται super-summit sets, είναι συζυγείς μέσω μίας permutation braid.

3.1 Επιθέσεις βασισμένες στο μήκος

Στην περίπτωση των πρωτοκόλλων που παρουσιάσαμε, δεν ψάχνουμε να βρούμε αν δύο λέξεις είναι συζυγείς, αλλά αυτό το γνωρίζουμε. Το πρόβλημα του επιτιθέμενου είναι δοθέντων δύο πλεξίδων $x, z = y^{-1}xy$, να βρει το y. Ο αλγόριθμος που χρησιμοποιούμε παρουσιάζεται στο [HT02].

Θεωρούμε μία συνάρτηση μήκους $l(x)$, όπως το μήκος της κανονικής μορφής. Αν υποθέσουμε ότι $y = y'\sigma_i$ με $l(y') < l(y)$, τότε για κάθε $j \neq i, l(\sigma_i z \sigma_i^{-1}) < l(\sigma_j z \sigma_j^{-1})$, γιατί $\sigma_i z \sigma_i^{-1} = \sigma_i y^{-1} x y \sigma_i^{-1} = \sigma_i \sigma_i^{-1} y'^{-1} x y' \sigma_i \sigma_i^{-1} = y'^{-1} x y'$.

Οπότε βρίσκουμε το σ_i και επαναλαμβάνουμε το ίδιο για το $z' = y'^{-1}xy'$ κ.ο.κ. Για να βρούμε το σ_i βρίσκουμε όλα τα $l(\sigma_j z \sigma_j^{-1})$ και παίρνουμε το μικρότερο. Αν είναι πολλά ίσα, τότε τα ελέγχουμε όλα.

Όσον αφορά στο χρόνο εκτέλεσής του αλγορίθμου, επειδή σε κάθε βήμα βρίσκει κάποιο σ_i της αναπαράστασης του y, τερματίζει σε χρόνο γραμμικό στο μήκος της μη κανονικής μορφής του y, σε αντίθεση με την εξαντλητική μέθοδο, που θα χρειαζόταν χρόνο εκθετικό στην κανονική μορφή του y, όπως προκύπτει από την παρατήρηση για την κανονική μορφή.

Ο αλγόριθμος δεν επιτυγχάνει πάντα και αυτό οφείλεται σε δύο λόγους.

Ο πρώτος είναι ότι επιλέγεται πάντα κάποιο j τέτοιο ώστε το μήκος του $l(\sigma_j z \sigma_j^{-1})$ να ελαχιστοποιηθεί. Το ότι το j αυτό θα είναι σωστό δεν είναι εξασφαλισμένο. Για να αντιμετωπιστεί το πρόβλημα αυτό θα έπρεπε να ελέγχονταν όλα τα j που ικανοποιούν την $l(y') < l(y)$, με $y = y'\sigma_i$, όμως τότε ο χρόνος εκτέλεσης του αλγορίθμου θα ήταν κοντά στον χρόνο εκτέλεσης του brute force αλγορίθμου.

Ο δεύτερος λόγος είναι ότι ο αλγόριθμος μπορεί να επιτύχει, δηλαδή να επιστρέψει κάποιο y' τέτοιο ώστε $y'^{-1}xy' = z$, όμως να μην ισχύει $y = y'$, όπως φαίνεται στο ακόλουθο παράδειγμα. Για $y = \sigma_1^{-4}$ και $g = \sigma_3^2\sigma_1\sigma_3$ ο αλγόριθμος επιστρέφει $y' = \sigma_3^4$ και ισχύει $y'^{-1}xy' = y^{-1}xy = \Delta^{-8}[2143][1243][1243]$, δηλαδή τα δύο αυτά στοιχεία έχουν την ίδια κανονική μορφή. Έτσι όταν χρησιμοποιήσουμε αυτήν την επίθεση για να βρούμε το κλειδί $a^{-1}b^{-1}gba = b^{-1}a^{-1}gab$, εάν έχουμε βρει ένα a' τ.ω. $a'^{-1}ga' = a^{-1}ga$, τότε υπάρχουν δύο περιπτώσεις:

Αν το a' περιέχει στοιχεία μόνο από το υποσύνολο του B_n από το οποίο επιτρέπεται να κατασκευαστεί το a τότε δεν υπάρχει πρόβλημα, αφού το σύνολο αυτό αντιμετατίθεται με το με στοιχεία του συνόλου από τα οποία επιτρέπεται να κατασκευαστεί το κλειδί b, άρα βρίσκουμε το σωστό κλειδί. Διαφορετικά, μπορεί να μην αντιμετατίθεται με το b, κι έτσι προκύπτει διαφορετικό κλειδί.

Εύκολα προκύπτει ότι η πιθανότητα επιτυχίας είναι τουλάχιστον $1/O(n^l)$, όπου l το μήκος του a, και αφού το μήκος αυτό είναι σταθερό, η πιθανότητα είναι $1/poly(n)$.

Τέλος, το ποσοστό επιτυχίας του εξαρτάται σημαντικά από την συνάρτηση μήκους που χρησιμοποιείται.

4 Υλοποίηση

Σε αυτήν την εργασία σχεδιάσαμε και υλοποιήσαμε αλγόριθμο εύρεσης κανονικής μορφής των πλεξίδων της ομάδας B_4 ($n = 4$). Στη συνέχεια υλοποιήσαμε τα πρωτόκολλα της ενότητας 2, πάνω σε αυτήν την ομάδα. Επίσης προσπαθήσαμε να υλοποιήσουμε επιθέσεις στο πρόβλημα της συζυγίας. Η μία επίθεση που επιχειρήσαμε είναι η εξαντλητική, ενώ η άλλη είναι αυτή που αναφέραμε στην προηγούμενη ενότητα (δηλ. την βασισμένη στο μήκος), χρησιμοποιώντας σαν συνάρτηση μήκους το Garside μήκος της λέξης.

4.1 Κανονική Μορφή

Για να υπολογίσουμε την Garside κανονική μορφή ενός στοιχείου $b = \sigma_1\sigma_2 \dots \sigma_n \in B_n$ χρειαζόμαστε για κάθε permutation να υπολογίσουμε το αρχικό και το τελικό τμήμα. Επιπλέον η κανονική μορφή έχει την μορφή $\Delta^u p_1 \dots p_k$, όπου p_i permutations. Για να μπορούμε να κάνουμε πράξεις μεταξύ στοιχείων αναπαριστούμε το στοιχείο Δ με το Braid $(\sigma_1 \dots \sigma_n)(\sigma_1 \dots \sigma_{n-1}) \dots \sigma_1$ και κάθε p_i με ένα μοναδικό braid. Για να μπορέσουμε να αντιστοιχίσουμε κάθε permutation με ένα μοναδικό braid καθώς και για να αποφύγουμε τον πολλαπλό υπολογισμό των αρχικών και τελικών τμημάτων χρησιμοποιούμε τον πίνακα *prmdata* στον οποίο οι πληροφορίες αυτές υπάρχουν.

Ο υπολογισμός της Garside κανονικής μορφής γίνεται από την συνάρτηση *normalform(n,b,prmdata)* που δέχεται σαν ορίσματα την παράμετρο *n*, μια αναπαράσταση *b* του στοιχείου της B_n και τον τον πίνακα *prmdata*. Η συνάρτηση αυτή αρχικά βρίσκει τον εκθέτη *ndelta* του Δ και το θετικό τμήμα *pos* του braid. Έπειτα, παράγει την αριστερή κανονική μορφή των permutations καλώντας την *leftnormal* και αναιρεί τις αρχικές εμφανίσεις του permutation [4 3 2 1] με τους εκθέτες του Δ. Τελικά, αντικαθιστά τον $\Delta^{-ndelta}$ και κάθε permutation με τα αντίστοιχα στοιχεία του B_n.

```
function [nf] = normalform(n,b,prmdata)

nf=[];
[pos,ndelta] = deltapos(n,b);
perm = left_normal(n,permutations(n,pos), prmdata);

while(perm(1,:)==[4 3 2 1])
    perm(1,:)=[];
    ndelta=ndelta+1;
end
d=inverse(delta(n));
for i=1:ndelta
    nf = [nf d];
end
[s,c]=size(perm);
for i = 1:s
    nf = [nf prmdata(permutation_number(n,perm(i,:))).braid];
end
```

Σαν παράδειγμα παραθέτουμε τον υπολογισμό της κανονικής μορφής του $\sigma_1\sigma_3^{-1}\sigma_2$:

```
>> normalform(n,[1 -3 2],prmdata)
pos =

```

```
     3     3     2     1     3     2     2
ndelta =

     1
perm =

     4     3     1     2
     2     3     1     4
nf =

    -1    -2    -1    -3    -2    -1     2     1     3     2     1     1     2
```

4.2 Ko *et al.* protocol

Το αρχείο cryptosystem.m υλοποιεί το πρωτόκολλο των Ko *et al.*. Αρχικά ανοίγει το αρχείο dataN3.mat για να πάρει την παράμετρο n και τον κατάλληλο πίνακα δεδομένων. Με τον τρόπο αυτό το σύστημα παραμετροποιείται έτσι ώστε να λειτουργεί για διάφορα n.

Έπειτα καθορίζονται οι παράμετροι του συστήματος, όπως το L, το επιθυμητό μέγεθος των μηκών των στοιχείων a, b, g, τα οποία παράγονται τυχαία και η σημαία των σχημάτων *dflag*, η οποία όταν τεθεί σε 1 εμφανίζει σχηματικά όλα τα braids που παράγονται κατά την διάρκεια του υπολογισμού. Τελικά γίνεται η κρυπτογράφηση και η αποκρυπτογράφηση, καλώντας τις αντίστοιχες συναρτήσεις. Παρατηρούμε ότι τα ορίσματα που δίνονται στις συναρτήσεις αποκρυπτογραφήσεις συμφωνούν με τις προδιαγραφές του συστήματος, δηλαδή, η συνάρτηση αποκρυπτογράφησης της Alice δέχεται σαν όρισμα το κρυφό κλειδί της Alice, *a*, η συνάρτηση αποκρυπτογράφησης του Bob δέχεται σαν όρισμα το κρυφό κλειδί του Bob, *b*, ενώ η συνάρτηση αποκρυπτογράφησης της Eve δέχεται σαν ορίσματα μόνο τα δημοσίως γνωστά δεδομένα.

```
%%data
s=open('dataN3.mat');
data=s.dataN3;

%%parameters
%if dflag=1 every braid will be dispayed in figure
dflag=0;
n=data.n;
prmdata=data.data;
L=round(n/2);
gLength=3;
aLenght=5;
bLenght=4;
g = randi(n-1,1,gLength);

%%encryption
[a,aga]=AliceEncrypt(data,L,g,aLenght,dflag);
[b,bgb]=BobEncrypt(data,L,g,bLenght,dflag);
%decryption
keya = AliceDencrypt(data,a,bgb,dflag);
keyb = BobDencrypt(data,b,aga,dflag);
keye = EveDencrypt(data,aga,bgb,g,dflag);
keye1 = EveDencrypt1(data,aga,bgb,g,dflag);
```

4.2.1 Συναρτήσεις Κρυπτογράφησης

Οι συναρτήσεις κρυπτογράφησης της Alice και του Bob είναι παρόμοιες, για τον λόγο αυτό παραθέτουμε μόνο της Alice.

Η Alice διαλέγει τυχαία ένα κλειδί (το οποίο τροποποιείται ώστε να αποτελείται από στοιχεία $\sigma_i, \in \{-(L-1), \ldots, -1, 1, \ldots, L-1\}$) και υπολογίζει την κανονική μορφή του στοιχείου $a^{-1}ga$ χρησιμοποιώντας την συνάρτηση *expbraid*, ο κώδικας της οποίας ακολουθεί. Τελικά επιστρέφονται το στοιχείο αυτό και το a, αφού είναι απαραίτητα για την αποκρυπτογράφηση.

```
function [res]=expbraid(base,expon)
    res=[inverse(expon) base expon];
```

```
function [a,aga] = AliceEncrypt(data,L,g,aLength,dflag)

n = data.n;
prmdata=data.data;

disp('Alice: chossing a...')
a = randi(n-3,1,aLength)+1;
ina = find(a>L-1);
a(ina)=a(ina)-2*L+1;

%calculating a^{-1}ga
aga = normalform(n,expbraid(g,a),prmdata);
```

4.2.2 Συναρτήσεις Αποκρυπτογράφησης

Οι συναρτήσεις αποκρυπτογράφησης της Alice και του Bob είναι παρόμοιες, για τον λόγο αυτό παραθέτουμε μόνο της Alice.

Στην συνάρτηση αποκρυπτογράφησης η Alice υπολογίζει το κλειδί της ως $k_a = a^{-1}b^{-1}gba$, όπου το $b^{-1}gb$ το δέχεται σαν όρισμα και επιστρέφει την κανονική του μορφή. Με τα νούμερα που χρησιμοποιούμε το κλειδί έχει ήδη μεγάλη έκταση, επιπλέον, το το αρχικό του τμήμα αποτελείται από κάποιο braid της μορφής Δ^{-k}, το οποίο δεν αυξάνει την πολυπλοκότητα του κλειδιού επειδή μπορεί εύκολα να αποκρυπτογραφηθεί. Για τον λόγο αυτό επιλέγουμε ως κλειδί να θεωρούμε μόνο το αρχικό τμήμα του k_a.

```
function [keya] = AliceDencrypt(data,a,bgb,dflag)
keya =  expbraid(bgb,a);
keya=normalform(data.n,keya,data.data);
keya=keya(find(keya>0));
```

4.2.3 Συναρτήσεις Επίθεσης

Έχουν υλοποιηθεί δύο συναρτήσεις επίθεσης και οι δύο ανάγονται στον υπολογισμό κάποιου ιδιωτικού κλειδιού (πχ. του a) δοσμένου του δημόσιου (πχ. του $a^{-1}ga$ και του g).

Η μια χρησιμοποιεί τον brute force τρόπο δοκιμάζοντας όλα τα πιθανά στοιχεία του B_n με αυξανόμενο μέγεθος. Ο κώδικας δεν έχει κάποιο ενδιαφέρον σημείο και για αυτό δεν θα τον παραθέσουμε.

Η δεύτερη χρησιμοποιεί τον αλγόριθμο που περιγράψαμε στην ενότητα 3.1 και η συνάρτηση αποκρυπτογράφησης παρατίθεται. Για τον υπολογισμό του ιδιωτικού κλειδιού χρησιμοποιείται η αναδρομική συνάρτηση *recDecrypt*. Επίσης έχει χρησιμοποιηθεί και συνάρτηση του Matlab που μετράει τον χρόνο που διαρκεί ο υπολογισμός του κλειδιού, ώστε να μπορέσουμε να τον συγκρίνουμε με αυτόν της brute force υλοποίησης.

```
function [keye] = EveDencrypt1(data,aga,bgb,g,dflag)
n=data.n;
prmdata=data.data;

%calculating a
t1=tic;
[flag,a]=recDecrypt(n, prmdata, aga, g, []);
time=toc(t1)

%calculating key
keye=[inverse(a) bgb a];
keye=normalform(data.n,keye,data.data);
keye=keye(find(keye>0));
```

Η συνάρτηση *recDecrypt* κατασκευάζει αναδρομικά το a. Συγκεκριμένα, αν το όρισμά της k είναι τέτοιο ώστε $a^{-1}ga = k^{-1}gk$ τότε επιστρέφει το k, διαφορετικά επιλέγω τα τα i που ελαχιστοποιούν το μήκος της $\sigma_i k a^{-1} g a k^{-1} \sigma_i^{-1}$. Αν το μήκος αυτό είναι μικρότερο του μήκους της $ka^{-1}gak^{-1}$, τότε αντικαθιστώ το k με $\sigma_i k$ και καλώ πάλι την συνάρτηση, διαφορετικά δηλώνω αποτυχία.

```
function [flag,k]=recDecrypt(n, prmdata, aga, g, k)

%given a^{-1}ga and a recursively calculates a

nf=normalform(n,[inverse(k) g k],prmdata);
nfh=aga;

if length(nfh)==length(nf)
    if all(nf==nfh)
        flag=1;
    else
        flag=0;
    end
else
    flag=0;
end

if flag==0
    h=[k aga inverse(k)];
    lh=lenE(h,n,prmdata);
    l0=lh+1;
    for i=-(n-1):-1
        c(i+n,:)=lenE([i h -i],n,prmdata);
    end
    for i=1:n-1
        c(i+n-1,:)=lenE([i h -i],n,prmdata);
    end
    [a,v]=min(c);
    if (a<l0)
```

```
        allv = find(c==a);
        for j=1:length(allv)
            v=allv(j);
            if (v<n)
                v=(v-n);
            else
                v=(v-n+1);
            end
        [b1,k]=recDecrypt(n,prmdata,aga,g,[v k]);
        if (b1==1)
            flag=1;
            return
        end
        end
    end
end
```

Παρατηρήσαμε ότι στις περισσότερες περιπτώσεις η συνάρτηση αυτή υπολογίζει το ιδιωτικό κλειδί τουλάχιστον 3 φορές γρηγορότερα από τον απλό brute force αλγόριθμο. Το μειονέκτημα της είναι ότι κάποιες φορές μπορεί να μην τερματίσει.

4.2.4 Παράδειγμα εκτέλεσης

Ακολουθεί ένα παράδειγμα εκτέλεσης του κώδικα.

```
>> cryptosystem
Ko et al. protocol

Encryption...
Alice: chossing a...
a=\sigma_{1}^{-4}
Alice: calculating a^{-1}ga...
a^{-1}ga=\sigma_{3}\sigma_{1}\sigma_{3}\sigma_{2}\sigma_{1}\sigma_{2}\sigma_{1}\sigma_{2}
   \sigma_{3}\sigma_{1}^{2}\sigma_{2}\sigma_{3}\sigma_{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}
   \sigma_{2}\sigma_{1}^{2}\sigma_{2}^{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}
Bob: chossing b...
b=\sigma_{3}^{4}
Bob: calculating b^{-1}gb...
b^{-1}gb=\sigma_{2}\sigma_{1}\sigma_{3}\sigma_{2}\sigma_{1}\sigma_{2}\sigma_{1}\sigma_{2}
        \sigma_{3}\sigma_{2}^{2}\sigma_{1}\sigma_{3}\sigma_{2}\sigma_{1}\sigma_{2}\sigma_{3}^{5}

Decryption...
Alice: calculating key...
key=a^{-1}(b^{-1}gb)a=\sigma_{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}\sigma_{1}\sigma_{3}
    \sigma_{2}\sigma_{1}^{2}\sigma_{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}\sigma_{1}\sigma_{3}
    \sigma_{2}\sigma_{1}\sigma_{2}^{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}\sigma_{1}^{2}
    \sigma_{2}^{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}^{5}
Bob: calculating key...
key=b^{-1}(a^{-1}ga)b=\sigma_{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}\sigma_{1}\sigma_{3}
    \sigma_{2}\sigma_{1}^{2}\sigma_{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}\sigma_{1}\sigma_{3}
    \sigma_{2}\sigma_{1}\sigma_{2}^{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}\sigma_{1}^{2}
    \sigma_{2}^{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}^{5}
Eve: calculating a with brute force...

time =
```

```
31
32    156.2322
33
34  Eve: calculating key...
35  key=\sigma_{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}\sigma_{1}\sigma_{3}\sigma_{2}
36      \sigma_{1}^{2}\sigma_{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}\sigma_{1}\sigma_{3}\sigma_{2}
37      \sigma_{1}\sigma_{2}^{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}\sigma_{1}^{2}
38      \sigma_{2}^{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}^{5}
39  Eve: calculating a...
40
41  time =
42
43     21.3587
44
45  Eve: calculating key...
46  key=\sigma_{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}\sigma_{1}\sigma_{3}\sigma_{2}
47      \sigma_{1}^{2}\sigma_{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}\sigma_{1}\sigma_{3}\sigma_{2}
48      \sigma_{1}\sigma_{2}^{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}\sigma_{1}^{2}
49      \sigma_{2}^{2}\sigma_{1}^{2}\sigma_{2}\sigma_{3}^{5}
```

4.3 Anshel *et al.* protocol

Το αρχείο cryptosystem2.m υλοποιεί το πρωτόκολλο των Anshel *et al.*. Όπως και στο cryptosystem.m αρχικά ανοίγει το αρχείο dataN3.mat για να πάρει την παράμετρο n και τον κατάλληλο πίνακα δεδομένων.

Έπειτα καθορίζονται οι παράμετροι του συστήματος, όπως τα k, m, a, b και η σημαία των σχημάτων *dflag*.

Τελικά γίνεται η κρυπτογράφηση και η αποκρυπτογράφηση, καλώντας τις αντίστοιχες συναρτήσεις. Όπως και στο προηγούμενο πρωτόκολλο, τα ορίσματα που δίνονται στις συναρτήσεις αποκρυπτογράφησης συμφωνούν με τις προδιαγραφές του συστήματος.

```
1  %%data
2  s=open('dataN3.mat');
3  data=s.dataN3;
4
5  %%parameters
6  %if dflag=1 every braid will be dispayed in figure
7  dflag=0;
8  n=data.n;
9  prmdata=data.data;
10 k=randi(10,1,1);
11 m=randi(10,1,1);
12 a=randi(2*n-2,k,3)-n+1;
13 b=randi(2*n-2,m,3)-n+1;
14
15 %%encryption
16 [xind,bx]=AliceEncrypt2(data,a,b,dflag);
17 [yind,ay]=BobEncrypt2(data,a,b,dflag);
18
19 %decryption
20 keya = AliceDencrypt2(data,a,b,xind,ay,dflag);
21 keyb = BobDencrypt2(data,a,b,yind,bx,dflag);
22 keye = EveDencrypt2(data,a,b,ay,bx,dflag);
```

4.4 Συναρτήσεις Κρυπτογράφησης

Οι συναρτήσεις κρυπτογράφησης της Alice και του Bob είναι παρόμοιες, για τον λόγο αυτό παραθέτουμε μόνο της Alice.

Η Alice διαλέγει τυχαία k στοιχεία $x_i, 1 \leq i \leq k$ και *xind* που είναι ο δείκτης που καθορίζει ότι το x_{xind} είναι το ιδιωτικό κλειδί της. Έπειτα, υπολογίζει την κανονική μορφή των στοιχείων $bx_j = x_{xind}^{-1} b_j x_{xind}, 1 \leq i \leq m$ χρησιμοποιώντας την συνάρτηση *expbraid*. Τελικά επιστρέφονται ο δείκτης *xind* και ένας πίνακας που περιέχει τα bx_j, αφού είναι απαραίτητα για την αποκρυπτογράφηση.

```
function [xind,bx] = AliceEncrypt2(data,a,b,dflag)

n = data.n;
prmdata=data.data;
k=size(a,1);

%chossing xind
xind=randi(k,1,1);
x=a(xind,:);
x=x(find(x));

%calculate bx
m=size(b,1);
sz=[];
for i=1:m
    sz(i) = size(normalform(n,[inverse(x) b(i,find(b(i,:))) x],prmdata),1);
end
bx=zeros(m,max(sz));
for i=1:m
    tmp=normalform(n,[inverse(x) b(i,find(b(i,:))) x],prmdata);
    bx(i,1:length(tmp))=tmp;
end
```

4.4.1 Συναρτήσεις Αποκρυπτογράφησης

Οι συναρτήσεις αποκρυπτογράφησης της Alice και του Bob είναι παρόμοιες, για τον λόγο αυτό παραθέτουμε μόνο της Alice.

Στην συνάρτηση αποκρυπτογράφησης η Alice υπολογίζει το κλειδί της πολλαπλασιάζοντας το στοιχείο του bx με δείκτη $xind$ με το x^{-1} και επιστρέφει το θετικό τμήμα της κανονικής του μορφής.

```
function [keya] = AliceDencrypt2(data,a,b,indx,ay,dflag)

x=a(indx,find(a(indx,:)));
keya = [inverse(x) ay(indx,find(ay(indx,:)))];
keya=normalform(data.n,keya,data.data);
keya=keya(find(keya>0));
```

4.5 Συνάρτηση Επίθεσης

Η συνάρτηση επίθεσης χρησιμοποιεί δύο φορές την συνάρτηση *recDecrypt*, μία για να υπολογίσει το x από τα b_1^x και το b_1 και μία για να υπολογίσει το y από τα a_1^y και a_1. Αν τα x και y είναι γνωστά, τότε το κλειδί μπορεί να υπολογιστεί εύκολα ως $key = x^{-1}y^{-1}xy$. Σημειώνουμε ότι όπως και στην περίπτωση της επίθεσης στο πρωτόκολλο των Ko *et al.* η επίθεση αυτή δεν επιτυγχάνει πάντα.

```
function [keye] = EveDencrypt2(data,a,b,ay,bx,dflag)
n=data.n;
prmdata=data.data;

%calculating x
[flagx,x]=recDecrypt(n, prmdata, bx(1,find(bx(1,:))), b(1,find(b(1,:))), []);

%calculating y
[flagy,y]=recDecrypt(n, prmdata, ay(1,find(ay(1,:))), a(1,find(a(1,:))), []);

keye = [inverse(x) inverse(y) x y ];
keye=normalform(data.n,keye,data.data);
keye=keye(find(keye>0));
```

4.6 Παράδειγμα Εκτέλεσης

Ακολουθεί ένα παράδειγμα εκτέλεσης του κώδικα.

```
>> cryptosystem2
Anshel et al. Protocol

Encryption...
Alice: chossing x...
x=a_{1}=\sigma_{1}^{-1}\sigma_{2}^{-1}
Alice: calculating b_i^{x}...
b_{1}^{x}=\sigma_{1}^{-1}
Bob: chossing y...
y=b_{1}=\sigma_{1}^{-1}
Bob: calculating a_i^{y}...
a_{1}^{y}=\sigma_{1}^{-1}\sigma_{2}^{-1}
a_{2}^{y}=\sigma_{2}^{-1}\sigma_{3}^{2}
a_{3}^{y}=\sigma_{1}^{-1}\sigma_{3}
a_{4}^{y}=\sigma_{1}^{-1}\sigma_{2}^{-1}
a_{5}^{y}=\sigma_{2}

Decryption...
Alice: calculating key...
key=\sigma_{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}
Bob: calculating key...
key=\sigma_{2}\sigma_{1}\sigma_{2}\sigma_{3}\sigma_{2}^{2}
Eve: calculating x...
Eve: calculating y...
Eve: calculating key...
```

5 Συμπεράσματα-Μελλοντική Εργασία

Στην εργασία αυτή, έγινε μία θεωρητική ανασκόπηση του πώς μη-μεταθετικές ομάδες και κυρίως η ομάδα των Braids μπορούν να χρησιμοποιηθούν για να υλοποιήσουν κρυπτογραφικά πρωτόκολλα. Επιπλέον παρουσιάστηκαν δύο τέτοια πρωτόκολλα καθώς και οι τρόποι επίθεσης σε αυτά.

Έπειτα παρουσιάστηκε μία υλοποίηση για τα δύο αυτά κρυπτοσυστήματα με την χρήση του Matlab. Τα δύο κύρια σημεία της υλοποίησης αφορούν την μετατροπή από και προς κανονική μορφή και την επίθεση.

Αρχικά τροποποιήσαμε τον αλγόριθμο της κανονικής μορφής και τον υλοποιήσαμε. Όπως αναφέραμε για να γίνει αποδοτικότερη η υλοποίηση αλλά και για να υπάρχει μοναδικός τρόπος αναπαράστασης ενός στοιχείου περιοριστήκαμε στην B_4. Σαν μελλοντική υλοποίηση θα μπορούσε να δοκιμαστεί το σύστημα για διαφορετικά n, ίσως ακόμα και να υλοποιηθεί για μεταβλητό n, αν και δεν υπάρχει προφανής τρόπος με τον οποίο ένα permutation να μπορεί μοναδικά να αντιστοιχηθεί σε κάποιο στοιχείο του B_n.

Όσον αφορά την επίθεση, ο αλγόριθμος που υλοποιήθηκε σπάει τα κρυπτοσυστήματα περίπου τρεις φορές γρηγορότερα από την εξαντλητική επίθεση. Και γενικότερα αυτός ο αλγόριθμος, είναι πολυωνυμικού χρόνου, όμως, για λόγους που αναπτύχθηκαν, δεν επιτυγχάνει πάντα. Στο σύστημα μας το ποσοστό επιτυχίας ήταν περίπου 1/4. Σαν μελλοντική εργασία, το ποσοστό αυτό θα μπορούσε να αυξηθεί, εντάσσοντας στους κώδικες περισσότερους ελέγχους ως προς την καταλληλότητα του κλειδιού που επιστρέφεται. Προφανείς βελτιώσεις είναι να πιστοποιείται να ελέγχεται ότι το κλειδί που επιστρέφεται ικανοποιεί τα κριτήρια που μπορεί να ελέγξει η Eve, όπως για παράδειγμα ότι έχει το σωστό μήκος ή αποτελείται από στοιχεία που ανήκουν στην κατάλληλη υποομάδα του B_n. Άλλη πρόταση είναι να επιστρέφονται πολλά κλειδιά και να ελέγχεται η εγκυρότητα όλων. Όμως χρειάζεται μελέτη για το ποιοι από τους ελέγχους αυτός αξίζει να προστεθούν, δηλαδή εισάγουν επιπλέον κόστος που ισοσταθμίζεται από ανάλογη μείωση της πιθανότητας αποτυχίας.

Γενικά, από τις παρατηρήσεις προκύπτει ότι η πιθανότητα επιτυχίας αλγορίθμου είναι $1/poly(n)$, αν το μήκος των ιδιωτικών κλειδιών είναι πολυωνυμικό ως προς το μήκος της αναn. Ένα τέτοιο ποσοστό, αν και είναι μικρό, στην κρυπτογραφία θεωρείται μη αμελητέο. Επομένως αυτά τα κρυπτοσυστήματα δεν είναι ασφαλή σε αυτή την μορφή τους, και είναι ανοιχτό πρόβλημα το αν μπορούν να τροποποιηθούν έτσι ώστε να αποδεικνύεται η ασφάλειά τους. Επίσης είναι ανοιχτό πρόβλημα η ασφάλειά τους, όταν χρησιμοποιούν άλλες μη μεταθετικές ομάδες, όπως αυτές που αναφέραμε στο θεωρητικό κομμάτι της εργασίας.

Αναφορές

[AAG99] Iris Anshel, Michael Anshel, and Dorian Goldfeld. An algebraic method for public-key cryptography. 1999.

[BCM09] Simon R. Blackburn, Carlos Cid, and Ciaran Mullan. Group theory in cryptography. June 2009.

[Bel07] James Belk. Thompson's group f, 2007.

[EM94] ELSAYED A. ELRIFAI and HUGH R. MORTON. Algorithms for positive braids. *Quarterly Journal of Mathematics*, 45:479–497, 1994.

[EPC+92] David B. A. Epstein, M. S. Paterson, G. W. Camon, D. F. Holt, S. V. Levy, and W. P. Thurston. *Word Processing in Groups.* A. K. Peters, Ltd., Natick, MA, USA, 1992.

[GAR] DAVID GARBER. Braid group cryptography preliminary draft.

[Gar69] H. Garside. The braid group and other groups. *Quarterly Journal of Mathematics*, 20:235–254, 1969.

[HS02] Dennis Hofheinz and Rainer Steinw. A practical attack on some braid group based cryptographic primitives. In *in Public Key Cryptography, 6th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2003 Proceedings, Y.G. Desmedt, ed., Lecture Notes in Computer Science 2567*, pages 187–198. Springer, 2002.

[HT02] J. Hughes and A. Tannenbaum. Length-based attacks for certain group based encryption rewriting systems, 2002.

[JWCP96] W. J. Floyd J. W. Cannon and W. R. Parry. Introductory notes on richard thompson's groups. In *L'Enseignement Mathematique*, 1996.

[KLC+00] Ki Hyoung Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju sung Kang, and Choonsik Park. New public-key cryptosystem using braid groups. In *Advances in cryptology—CRYPTO 2000 (Santa Barbara, CA), 166–183, Lecture Notes in Comput. Sci. 1880*, pages 166–183. Springer, 2000.

[MK06] Francesco Matucci Martin Kassabov. The simultaneous conjugacy problem in thompson's group f. Cornall University Library, 2006.