ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ,
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

# ΑΣΦΑΛΗΣ ΔΙΑΣΥΝΔΕΣΗ ΕΓΓΡΑΦΩΝ ΚΑΤΑΝΕΜΗΜΕΝΩΝ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ ΜΕ ΧΡΗΣΗ ΠΡΩΤΟΚΟΛΛΩΝ ΚΡΥΠΤΟΓΡΑΦΗΣΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΜΗΤΡΟΓΙΑΝΝΗΣ ΒΑΣΙΛΕΙΟΣ

ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ: ΒΑΣΙΛΕΙΟΣ ΒΕΡΥΚΙΟΣ
ΗΛΙΑΣ ΧΟΥΣΤΗΣ
ΠΑΝΑΓΙΩΤΗΣ ΜΠΟΖΑΝΗΣ

Ιούλιος 2008

# Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά:

- Τους επιβλέποντες καθηγητές κ.κ. Β. Βερύκιο, Η. Χούστη και Π. Μποζάνη για τη άψογη συνεργασία και υποστήριξη τους σε όλα τα στάδια της εργασίας, για την απαραίτητη καθοδήγηση που μου παρείχαν, καθώς και για τις καθοριστικές συμβουλές τους σε όλα τα κρίσιμα ζητήματα.
- Την οικογένεια μου για την υποστήριξη και συμπαράσταση τους καθόλη την διάρκεια της εργασίας.

# Περιεχόμενα

# Λίστα Πινάκων

# Λίστα Σχημάτων

# Αντιστοίχιση Ελληνικής Ορολογίας στην Αγγλική

| Ελληνικός Όρος | Αγγλικός όρος |
|---|---|
| Ακεραιότητα των δεδομένων | Data Integrity |
| Αλληλουχία | Concatenation |
| Αλφαριθμητικός χαρακτήρας | String |
| Αναγνωριστής | Identifier |
| Ανάθεση ετικετών | Tag assignment |
| Ανάκτηση πληροφορίας | Information Retrieval |
| Αντιμετάθεση | Transposition |
| Δημιουργία συνόλων δεδομένων | Data Set Generation |
| Δημόσιο ή Ασύμμετρο Κλειδί | Public Key |
| Διαμοίραση δεδομένων | Data Sharing |
| Διασύνδεση εγγραφών | Record Linkage |
| Διατήρηση της εμπιστευτικότητας κατά την εξόρυξη δεδομένων | Privacy-preserving Data Mining |
| Διαχωριστικό | Separator |
| Διερμηνευτής | Interpreter |
| Διπλότυπο | Duplicate |
| Δυναμό-σύνολο | Power-set |
| Εκχωρημένος | Assigned |
| Εμπιστευτικότητα των πληροφοριών | Information Privacy |
| Εμφωλιασμένο | Nested |
| Ενοποίηση των δεδομένων | Data Integration |
| Ένταση της Διαφοράς | Magnitude of Difference |
| Επίθεμα | Suffix |
| Καθαρισμός και Τυποποίηση εγγραφών | Record Cleaning and Standardization |
| Κανόνας απόφασης | Decision rule |
| Κατανεμημένη Βάση Δεδομένων | Distributed Database |
| Κατώφλι | Threshold |
| Μέθοδος Δέσμης | Bind Method |
| Μέθοδος Ροής | Flow Method |
| Μεταγλωττιστής | Compiler |
| Μετατοπιζόμενο παράθυρο | Sliding window |
| Μετρητής | Counter |
| Μετρικά Απόστασης | Distance Metrics |
| Μυστικό ή Συμμετρικό Κλειδί | Private or Symmetric Key |

| | |
|---|---|
| Ν-γράμμα | N-gram |
| Οντότητα | Entity |
| Πίνακας κατακερματισμού | Hash table |
| Πρόθεμα | Prefix |
| Συνάθροιση | Aggregation |
| Συνάρτηση μονόδρομου κατακερματισμού | One-way Hash function |
| Συνένωση των δεδομένων | Data Join |
| Σύνολο αντικειμένων | Item-set |
| Σύνοψη μηνύματος | Message Digest |
| Ταίριασμα των αντικειμένων | Object Matching |
| Ταίριασμα των σχημάτων | Schema Matching |
| Ταξινομημένη Γειτνίαση | Sorted Neighborhood |
| Τμηματοποίηση | Segmentation |
| Υπο-λίστα | Sub-list |
| Υπό-ρουτίνα | Subroutine |
| Χαρακτήρας οριοθέτησης | Delimiter |

# Περίληψη

Η ασφαλής διασύνδεση εγγραφών είναι μια πολύ σημαντική διαδικασία, ιδιαίτερα όταν εφαρμόζεται για την επεξεργασία ευαίσθητων προσωπικών δεδομένων. Οι διάφοροι οργανισμοί προσπαθούν να συγκρίνουν την ομοιότητα των δεδομένων τους, χωρίς όμως να αποκαλύπτονται ιδιωτικές ή ανταγωνιστικές πληροφορίες σε τρίτους. Προς αυτήν την κατεύθυνση, διάφορες επιστημονικές ομάδες έχουν αναπτύξει και προτείνει αρκετές μεθόδους και πρωτόκολλα που πληρούν αυτές τις προϋποθέσεις.

Η διαδικασία της διασύνδεσης των εγγραφών ουσιαστικά συγκρίνει τις εγγραφές που υπάρχουν σε δυο ή περισσότερα σύνολα δεδομένων, και εξετάζει κατά πόσο κάποιες εγγραφές αντιστοιχούν στην ίδια οντότητα του πραγματικού κόσμου. Σε αυτήν την εργασία γίνεται η διερεύνηση ορισμένων μεθόδων που επιτυγχάνουν την ασφαλή διασύνδεση των γνωρισμάτων των εγγραφών [6, 10, 15, 21]. Ουσιαστικά η μέθοδος εφαρμόζεται ξεχωριστά για κάθε γνώρισμα των εγγραφών και κατόπιν, συγκρίνοντας την ομοιότητα των γνωρισμάτων, κρίνεται αν οι εγγραφές ταιριάζουν ή όχι. Για την επίτευξη της ασφαλούς διασύνδεσης εγγραφών θα προταθούν τέσσερις τεχνικές, καθώς και ορισμένες βελτιώσεις τους, οι οποίες βασίζονται στο ίδιο πλαίσιο ασφαλούς ανταλλαγής πληροφοριών, το οποίο είναι ευρύτερα γνωστό ως μοντέλο τρίτης έμπιστης πηγής [20]. Οι τέσσερις τεχνικές υλοποιούνται υπό το πλαίσιο τεσσάρων πρωτοκόλλων. Ένα πρωτόκολλο είναι ένα σύνολο τυπικών κανόνων που περιγράφουν πώς οφείλει να εκτελείται μια ενέργεια.

Δύο από τα πρωτόκολλα που θα παρουσιαστούν στις ενότητες 3.2 και 3.3 υλοποιούν αλγορίθμους σύγκρισης της ομοιότητας των δεδομένων με χρήση N-γραμμάτων [10, 21, 23]. Συγκεκριμένα στο πρώτο παράγονται N-γράμματα καθώς και τα δυναμοσύνολα τους [10]. Μάλιστα προτείνεται και μια βελτιωμένη εκδοχή του όπου παράγονται μόνο δύο υπο-λίστες N-γραμμάτων και όχι ολόκληρο το δυναμοσύνολο.

Στο δεύτερο πρωτόκολλο παράγονται μόνο Τρι-γράμματα [23] στα οποία προστίθεται ένα διάνυσμα που περιλαμβάνει αριθμούς που αντιστοιχούν στο πόσες φορές εμφανίζεται το αντίστοιχο Τρι-γράμμα μέσα στο αλφαριθμητικό.

Τα άλλα δύο πρωτόκολλα που παρουσιάζονται στις ενότητες 3.4 και 3.5 υλοποιούν αλγορίθμους σύγκρισης της ομοιότητας με χρήση μετρικών απόστασης. Το ένα χρησιμοποιεί την απόσταση Levenshtein [35], ενώ το άλλο εφαρμόζει την απόσταση Jaro – Winkler [46, 47]. Σε όλα τα πρωτόκολλα η ασφάλεια επιτυγχάνεται κάνοντας χρήση κρυπτογραφικών τεχνικών και συναρτήσεων κατακερματισμού.

Παράλληλα, προκειμένου να μειωθεί ο χρόνος εκτέλεσης των πρωτοκόλλων, υλοποιήθηκε μια εναλλακτική εκδοχή όλων των πρωτοκόλλων όπου εφαρμόζεται στα δεδομένα μια τεχνική Blocking [1, 3, 4, 28]. Με την τεχνική αυτή οι εγγραφές αρχικά οργανώνονται σε μπλοκς με βάση ένα ή περισσότερα κοινά γνωρίσματα, και η σύγκριση γίνεται μόνο μεταξύ εγγραφών που ανήκουν στο ίδιο μπλοκ.

Για όλα τα πρωτόκολλα υλοποιήθηκαν ορισμένα προγράμματα στην γλώσσα προγραμματισμού Perl [56]. Η παρουσίαση, καθώς και επεξήγηση του πηγαίου κώδικα των προγραμμάτων γίνεται στην εργασία.

Τέλος, παρατίθενται ορισμένα ενδεικτικά πειραματικά αποτελέσματα εκτέλεσης των πρωτοκόλλων. Από αυτά εξάγονται χρήσιμα συμπεράσματα όσον αφορά την συνολική απόδοση των πρωτοκόλλων, τα οποία αφορούν τους χρόνους εκτέλεσης τους και την ικανότητα τους να επιτυγχάνουν με ορθό τρόπο και με ασφάλεια την διασύνδεση των εγγραφών.

# Κεφάλαιο 1

# Εισαγωγή

Μια ιδιαίτερα σημαντική διεργασία του ευρύτερου τομέα των Βάσεων Δεδομένων είναι η ενοποίηση των δεδομένων από πολλές διαφορετικές πηγές [11, 12]. Η συγκεκριμένη διαδικασία όταν εφαρμόζεται με ορθό τρόπο μπορεί να προσφέρει αρκετά οφέλη σε πληθώρα εφαρμογών, όπως είναι η κοινή χρήση επιστημονικών και ιατρικών δεδομένων, καθώς και η ανταλλαγή πληροφοριών μεταξύ κυβερνητικών οργανισμών ή ανταγωνιστικών επιχειρήσεων [11]. Γενικότερα οι οργανισμοί διαθέτουν πολύ μεγάλο όγκο πληροφοριών που είναι οργανωμένες στις Βάσεις Δεδομένων τους. Πολλές φορές διαφορετικές εγγραφές που περιέχονται σε μια ή περισσότερες Βάσεις Δεδομένων, αντιστοιχούν στην ίδια οντότητα, με αποτέλεσμα να αυξάνεται το πλήθος των δεδομένων χωρίς λόγο, και να δυσχεραίνεται η σωστή διαχείριση και επεξεργασία των Βάσεων Δεδομένων. Επομένως η ενοποίηση των δεδομένων απαιτεί την ανίχνευση και απαλοιφή όλων των διπλότυπων εγγραφών.

Η ενοποίηση των δεδομένων είναι δυνατόν να επιτευχθεί μέσα από την διαδικασία της διασύνδεσης εγγραφών. Ουσιαστικά με τον όρο διασύνδεση εγγραφών εννοούμε την διαδικασία κατά την οποία εξετάζουμε αν δύο εγγραφές από διαφορετικά σύνολα δεδομένων αντιστοιχούν στην ίδια οντότητα του πραγματικού κόσμου [6, 10, 15].

Ένας ανασταλτικός παράγων που αποτρέπει τους διάφορους οργανισμούς από το να προβούν στην παραπάνω διαδικασία, είναι τα προβλήματα ασφάλειας που προκύπτουν κατά την ανταλλαγή των πληροφοριών. Για παράδειγμα, προκειμένου να επιτευχθεί η ανταλλαγή ιατρικών επιστημονικών δεδομένων πρέπει πρώτα να γίνει η ενοποίηση των δεδομένων αυτών. Όμως κατά την εξέλιξη αυτής της διαδικασίας, ελλοχεύει ο κίνδυνος να διαρρεύσουν ευαίσθητα προσωπικά δεδομένα ασθενών σε μη

εξουσιοδοτημένα άτομα. Για τον λόγο αυτό, η επιστημονική κοινότητα στρέφεται προς την ανάπτυξη μεθόδων που θα επιτρέπουν την ανταλλαγή και ενοποίηση των δεδομένων, διατηρώντας παράλληλα την ακεραιότητα και την εμπιστευτικότητα των πληροφοριών που αυτά περιέχουν.

Προκειμένου λοιπόν να επιτευχθεί ο στόχος της ενοποίησης των δεδομένων διατηρώντας την εμπιστευτικότητα των πληροφοριών, πολλοί ερευνητές αναπτύσσουν πρωτόκολλα που συνδυάζουν μεθόδους διασύνδεσης εγγραφών με διαδικασίες που μπορούν να εγγυηθούν την ασφαλή ανταλλαγή και επεξεργασία των δεδομένων.

## 1.1 Γενικό Υπόβαθρο

Η συγκεκριμένη εργασία αποτελεί ουσιαστικά συνέχεια προηγούμενης μελέτης [51] πάνω στην ασφαλή ενοποίηση δεδομένων διαφορετικών Βάσεων Δεδομένων. Η ασφαλής ενοποίηση των δεδομένων είναι μια διαδικασία η οποία αποτελείται από τρία κυρίως βήματα, ένα εκ των οποίων είναι η διασύνδεση των εγγραφών.

- Το πρώτο βήμα είναι ο καθαρισμός και η τυποποίηση των δεδομένων [9]. Όταν εξετάζονται δύο σύνολα δεδομένων από διαφορετικές πηγές υπάρχει η πιθανότητα να μην βρίσκονται στην ίδια τυποποιημένη μορφή, όπως για παράδειγμα το ένα σύνολο να περιλαμβάνει στο ίδιο πεδίο το όνομα και το επώνυμο, ενώ το άλλο σύνολο να διαθέτει ξεχωριστά πεδία για αυτά τα χαρακτηριστικά. Για το λόγο αυτό εφαρμόζονται τεχνικές που μετατρέπουν τα δύο σύνολα δεδομένων στην ίδια τυποποιημένη μορφή.
- Στο επόμενο βήμα εφαρμόζεται η διαδικασία της απαλοιφής των διπλότυπων [9], κατά την οποία κάθε σύνολο εγγραφών εντοπίζει τις εγγραφές που αντιστοιχούν στην ίδια οντότητα και απαλείφει τις διπλότυπες. Ο τρόπος λειτουργίας της απαλοιφής των διπλότυπων είναι όμοιος με αυτόν της διασύνδεσης των εγγραφών, μόνο που εφαρμόζεται μεταξύ εγγραφών του ίδιου συνόλου δεδομένων, και είναι απαραίτητη προκειμένου να μειωθεί ο όγκος της

πληροφορίας που ανταλλάσσουν και επεξεργάζονται οι συνεργαζόμενες Βάσεις Δεδομένων.

- Τέλος, το τρίτο βήμα είναι η διαδικασία της διασύνδεσης των εγγραφών.

Το σημαντικότερο ίσως τμήμα της παραπάνω διαδικασίας είναι η διασύνδεση των εγγραφών. Η συγκεκριμένη εργασία εξετάζει ορισμένες τεχνικές που επιτυγχάνουν την ορθή διασύνδεση. Συγκεκριμένα, κάθε μια από αυτές τις τεχνικές εφαρμόζει διασύνδεση των γνωρισμάτων κάθε ζεύγους εγγραφών. Κατόπιν, έχοντας συγκρίνει έναν ικανοποιητικό αριθμό ζευγών γνωρισμάτων, που ενδέχεται να είναι από ένα ως όλα τα γνωρίσματα των εγγραφών, είμαστε σε θέση να κρίνουμε αν οι εγγραφές ταιριάζουν ή όχι. Δηλαδή η επιτυχής διασύνδεση γνωρισμάτων στο σύνολο της, οδηγεί στην επιτυχή διασύνδεση εγγραφών. Προχωρώντας ένα βήμα πιο μπροστά, διερευνά μια εναλλακτική προσέγγιση του θέματος, στην οποία κάθε σύνολο δεδομένων πρώτα από όλα οργανώνει τις εγγραφές του σε μπλοκ [3] με βάση κοινά χαρακτηριστικά, και κατόπιν εφαρμόζει την διασύνδεση. Κατά αυτόν τον τρόπο επιτυγχάνεται η μείωση του πλήθους των συγκρίσεων και ακολούθως ο φόρτος. Τα παραπάνω συνδυάζονται με τεχνικές κρυπτογράφησης [20] ώστε όλη η διαδικασία να βρίσκεται πάντοτε υπό το πλαίσιο της ιδιωτικότητας των πληροφοριών.

## 1.2 Κεντρική Ιδέα Εργασίας

Όπως αναφέρθηκε στην ενότητα 1.1, ο στόχος της συγκεκριμένης έρευνας είναι η εξέταση και ανάλυση ορισμένων τεχνικών οι οποίες μπορούν να φέρουν εις πέρας την διαδικασία της ασφαλούς διασύνδεσης εγγραφών με επιτυχία. Γενικά, τα θέματα στα οποία εστιάζεται το ενδιαφέρον της έρευνας μπορούν να συνοψιστούν ως εξής:

- Γίνεται η παρουσίαση και υλοποίηση τεσσάρων τεχνικών που επιτυγχάνουν την διασύνδεση γνωρισμάτων. Οι τεχνικές αυτές είναι η χρήση N-γραμμάτων με δημιουργία δυναμό-συνόλων [10], Τρι-γραμμάτων [23], της απόστασης Levenshtein [35] και της απόστασης Jaro-Winkler [46, 47]. Οι παραπάνω τεχνικές επιτυγχάνουν την σύγκριση της ομοιότητας δύο αλφαριθμητικών.

- Στη κατεύθυνση της βελτίωσης της απόδοσης των παραπάνω τεχνικών, εξετάζεται μια εναλλακτική προσέγγιση όπου, προτού γίνει η σύγκριση, οι εγγραφές οργανώνονται σε μπλοκ με βάση ένα χαρακτηριστικό ομοιότητας.
- Στις τεχνικές διασύνδεσης γνωρισμάτων που αναφέρθηκαν προηγουμένως εφαρμόζονται συγκεκριμένοι αλγόριθμοι κρυπτογράφησης [16, 37], και συναρτήσεις κατακερματισμού [20], ώστε να εξεταστεί κατά πόσο οι τεχνικές είναι σε θέση να επιτύχουν την διασύνδεση εγγραφών χωρίς παράλληλα να υπάρχει διαρροή ευαίσθητων πληροφοριών.
- Προκειμένου να επιτευχθεί η μέτρηση της απόδοσης των παραπάνω τεχνικών, λαμβάνουν χώρα και παρουσιάζονται αρκετά πειράματα στα οποία χρησιμοποιήθηκαν πολλά διαφορετικά σύνολα δεδομένων, τόσο ως προς το πλήθος των εγγραφών τους, αλλά και ως προς το είδος των αλφαριθμητικών που αυτές περιείχαν.

## 1.3 Συνεισφορά της Εργασίας

Η εργασία κινείται στα πλαίσια ενός τομέα των Βάσεων Δεδομένων που αφορά την ασφαλή ενοποίηση των δεδομένων. Ο τομέας αυτός τα τελευταία χρόνια έχει επικεντρώσει το ενδιαφέρον αρκετών ερευνητών και παρουσιάζει ιδιαίτερη ανάπτυξη. Η συνεισφορά λοιπόν της εργασίας έγκειται κυρίως στα παρακάτω:

- **Θεωρητική Ανάλυση Πρωτοκόλλων Ασφαλούς Διασύνδεσης Γνωρισμάτων:** Επιτυγχάνεται η δημιουργία τεσσάρων πρωτοκόλλων ασφαλούς διασύνδεσης γνωρισμάτων, στα οποία εφαρμόζονται οι τεχνικές που προαναφέρθηκαν στην ενότητα 1.2. Γίνεται λεπτομερής παρουσίαση όλων των βημάτων που ακολουθούνται, καθώς και των βασικών κοινών γνωρισμάτων που έχουν όλα τα πρωτόκολλα. Για κάθε ένα πρωτόκολλο γίνεται ανάλυση όλων των παραμέτρων που πρέπει να ληφθούν υπόψιν.
- **Παρουσίαση Τεχνικών Blocking:** Παρουσιάζονται διάφορες τεχνικές δημιουργίας μπλοκ εγγραφών οι οποίες επιτυγχάνουν σημαντική μείωση του

κόστους σε υπολογιστικούς πόρους κατά την σύγκριση των εγγραφών. Από τις τεχνικές Blocking που παρουσιάζονται, εφαρμόζεται η πιο ενδεδειγμένη για τα παραπάνω πρωτόκολλα και εξετάζεται κατά πόσο βελτιώνει όντως την απόδοση τους και ποιες είναι οι συνέπειες της εφαρμογής της στο ταίριασμα όλων των όμοιων εγγραφών.

- **Τεχνικές Κρυπτογράφησης:** Εξετάζεται κατά πόσο η εφαρμογή των αλγόριθμων κρυπτογράφησης είναι σε θέση να επιτύχει την ασφαλή διασύνδεση εγγραφών. Παράλληλα ελέγχεται αν επιβαρύνουν σε σημαντικό βαθμό το υπολογιστικό κόστος εκτέλεσης των πρωτοκόλλων.
- **Δημιουργία Βιβλιοθήκης Προγραμμάτων:** Υλοποιούνται συνολικά 20 προγράμματα. Αυτά είναι στη διάθεση των χρηστών να τα εκτελέσουν και να εξετάσουν την απόδοση των πρωτοκόλλων θέτοντας τις δικές τους παραμέτρους. Επίσης δίνεται η δυνατότητα ένταξης των συγκεκριμένων προγραμμάτων σε ένα ολοκληρωμένο σύστημα διασύνδεσης εγγραφών, όπως είναι το Febrl [9], ώστε ο χρήστης του συστήματος να διαθέτει περισσότερες εναλλακτικές λύσεις επεξεργασίας των δεδομένων του και πειραματισμού.
- **Πειραματικά Αποτελέσματα:** Γίνονται επαναληπτικές εκτελέσεις των προγραμμάτων χρησιμοποιώντας κάθε φορά διαφορετικές παραμέτρους, ώστε να είναι διαθέσιμα αρκετά πειραματικά αποτελέσματα. Η παρουσίαση τους αποδεικνύει για κάθε πρωτόκολλο ξεχωριστά, κατά πόσο μπορεί να επιτύχει την ασφαλή διασύνδεση εγγραφών και σε τι χρονικά πλαίσια.

## 1.4 Δομή της Εργασίας

Η δομή της υπόλοιπης εργασίας έχει ως εξής:

- Στο δεύτερο κεφάλαιο περιγράφονται ορισμένες χρήσιμες βασικές έννοιες που εμφανίζονται στην εργασία. Οι έννοιες αυτές αφορούν τις τεχνικές που χρησιμοποιούνται για την διασύνδεση εγγραφών, τις διάφορες τεχνικές

δημιουργίας μπλοκ εγγραφών, καθώς και ορισμένους κρυπτογραφικούς ορισμούς.

- Στο τρίτο κεφάλαιο παρουσιάζονται τα πρωτόκολλα που εφαρμόζουν τις τεχνικές χρήσης N-γραμμάτων, Τρι-γραμμάτων, και των αποστάσεων Levenshtein και Jaro-Winkler. Αρχικά παρατίθενται τα τέσσερα πρωτόκολλα ξεχωριστά και στη συνέχεια η εναλλακτική εκδοχή τους που κάνει χρήση του blocking.
- Στο τέταρτο κεφάλαιο παρουσιάζεται η υλοποίηση και ο σχολιασμός των προγραμμάτων που αναπτύχθηκαν για τους αλγόριθμους που επιτυγχάνουν την ασφαλή διασύνδεση των εγγραφών, όπως επίσης και για τις block εκδόσεις των παραπάνω αλγόριθμων.
- Στο πέμπτο κεφάλαιο λαμβάνει χώρα η αξιολόγηση των αλγόριθμων που αναπτύχθηκαν στα πλαίσια του τέταρτου κεφαλαίου. Σε αυτή συμπεριλαμβάνονται τόσο τα χρονικά αποτελέσματα της εκτέλεσης των πειραμάτων, όσο και τα ποσοστά επιτυχούς ταιριάσματος των εγγραφών που αντιστοιχούν στην ίδια οντότητα.
- Τέλος στο έκτο κεφάλαιο δίνεται ο επίλογος της εργασίας, όπου παρατίθενται τα τελικά συμπεράσματα, αλλά και οι μελλοντικοί στόχοι της συγκεκριμένης έρευνας.

# Κεφάλαιο 2

## Βασικές Έννοιες

Στο παρόν κεφάλαιο θα γίνει η παρουσίαση ορισμένων βασικών εννοιών που θα συναντήσουμε μέσα στην εργασία. Η κατανόηση αυτών των εννοιών κρίνεται απαραίτητη προκειμένου κάποιος να είναι σε θέση να κατανοήσει τα ζητήματα που μελετάμε. Οι έννοιες οι οποίες περιγράφονται πρώτα σχετίζονται με την διαδικασία της διασύνδεσης εγγραφών. Ουσιαστικά αποτελούν τις διάφορες τεχνικές που εφαρμόσθηκαν στα πρωτόκολλα χρήσης N-γραμμάτων, Τρι-γραμμάτων, και των αποστάσεων Levenshtein και Jaro-Winkler, προκειμένου να επιτευχθεί η διασύνδεση των γνωρισμάτων. Αρχικά πρέπει να δοθεί ένας σύντομος ορισμός μερικών εννοιών που συναντάμε στην εργασία. Όπως προαναφέρθηκε στην περίληψη, πρωτόκολλο είναι ένα σύνολο από τυπικούς κανόνες που περιγράφουν τον επιτρεπτό τρόπο εκτέλεσης μιας ενέργειας, όπως είναι μια τεχνική διασύνδεσης εγγραφών. Επίσης, ένας αλγόριθμος είναι το σύνολο των κανόνων που εφαρμόζονται για την επίλυση ενός συγκεκριμένου προβλήματος. Εκτός των άλλων, μια διαδικασία είναι η μεθοδευμένη σειρά ενεργειών που οδηγούν σε ένα ορισμένο αποτέλεσμα.

Εν συνεχεία γίνεται μια σύντομη παρουσίαση ορισμένων βασικών μεθόδων δημιουργίας μπλοκ εγγραφών. Το τελευταίο τμήμα του κεφαλαίου αποτελείται από την επεξήγηση των εννοιών που αφορούν την διατήρηση της ασφάλειας, δηλαδή περιγράφονται οι κρυπτογραφικές τεχνικές που εφαρμόσθηκαν.

## 2.1 Τεχνικές Διασύνδεσης Εγγραφών

Όπως έχει προαναφερθεί στην εισαγωγή του κεφαλαίου 1, η διασύνδεση εγγραφών [6, 10, 15] είναι η διαδικασία συγκρίσεως εγγραφών προκειμένου να αποφασισθεί κατά πόσο αυτές ταιριάζουν ή δεν ταιριάζουν. Δηλαδή αν αναπαριστούν την ίδια ή τελείως διαφορετικές οντότητες. Εάν το σύστημα διασύνδεσης εγγραφών δεν είναι σε θέση να πάρει κάποια απόφαση, τότε είναι απαραίτητη η ανθρώπινη παρέμβαση ώστε να αποφασισθεί αν πράγματι ταιριάζουν ή όχι.

Στην βιβλιογραφία η διασύνδεση εγγραφών ενδέχεται να αναφέρεται με διαφορετικούς όρους. Συχνά εμφανίζεται ως «ταίριασμα δεδομένων» ή «πρόβλημα αναγνώρισης αντικειμένου» [22]. Επίσης ονομάζεται «διαδικασία συγχώνευσης/ εκκαθάρισης» και «επικάλυψη λιστών» [9]. Εκτός των άλλων στην περίπτωση που η διαδικασία εφαρμόζεται μεταξύ εγγραφών που ανήκουν στο ίδιο σύνολο δεδομένων, τότε ονομάζεται «απαλοιφή των διπλότυπων» [9]. Όλοι οι παραπάνω διαφορετικοί ορισμοί της διασύνδεσης των εγγραφών αναφέρονται πάντα στην ίδια διαδικασία.

Η διασύνδεση εγγραφών, ανάλογα με την περίσταση, μπορεί να είναι από απλοϊκή ως ιδιαίτερα πολύπλοκη. Στην περίπτωση όπου για κάθε οντότητα υφίσταται ένας μοναδικός αναγνωριστής, που την κάνει να ξεχωρίζει από όλες τις υπόλοιπες οντότητες, και ο οποίος θα είναι ίδιος σε όλα τα σύνολα δεδομένων, τότε η διαδικασία είναι πολύ απλή και μπορεί να περιοριστεί στην εφαρμογή κάποιας τεχνικής που κάνει συνένωση των συνόλων δεδομένων. Αντίθετα, εάν δεν υπάρχει κάποιος αναγνωριστής, τότε πρέπει να χρησιμοποιηθούν συγκεκριμένες τεχνικές διασύνδεσης. Ορισμένες από αυτές είναι οι τεχνικές που χρησιμοποιούν N-γράμματα [10, 21], οι μέθοδοι που κάνουν χρήση της απόστασης Levenshtein [35], καθώς και οι τεχνικές που εφαρμόζουν την απόσταση Jaro – Winkler [46, 47].

### 2.1.1 Τεχνικές Χρήσης Ν-γραμμάτων

Οι τεχνικές χρήσης Ν-γραμμάτων [10, 21] στην ουσία αποτελούν μεθόδους σύγκρισης της ομοιότητας μεταξύ δύο αλφαριθμητικών χαρακτήρων, ή ακόμη και μεταξύ ολόκληρων ακολουθιών τέτοιων αλφαριθμητικών [23]. Επομένως οι συγκεκριμένες τεχνικές είναι δυνατόν να χρησιμοποιηθούν για την σύγκριση των αλφαριθμητικών που περιέχονται ξεχωριστά σε κάθε πεδίο των εγγραφών. Μάλιστα είναι σε θέση να αντιμετωπίσουν περιπτώσεις, όπου ένα πεδίο περιλαμβάνει περισσότερα από ένα αλφαριθμητικά.

Με τον όρο Ν-γράμμα εννοείται ότι ένα αλφαριθμητικό διαχωρίζεται σε ακολουθίες από Ν χαρακτήρες. Δηλαδή αν Ν=2, οπότε έχουμε ένα δι-γράμμα, το αλφαριθμητικό χωρίζεται σε ακολουθίες των δύο χαρακτήρων. Αντίστοιχα αν Ν=3 έχουμε ένα τρι-γράμμα που αποτελείται από ακολουθίες τριών χαρακτήρων. Προκειμένου να γίνει το παραπάνω ευκολότερα κατανοητό θα γίνει διαχωρισμός σε δι-γράμματα της λέξεως 'Ιωάννης'. Τότε υφίστανται τα εξής δι-γράμματα:

'Ιωάννης': ('ιω', 'ωα', 'αν', 'νν', 'νη', 'ης')

Αντίστοιχα για την περίπτωση όπου γίνεται χρήση τρι-γραμμάτων τότε δημιουργούνται τα παρακάτω:

'Ιωάννης': ('ιωα', 'ωαν', 'ανν', 'ννη', 'νης')

Όπως φαίνεται από τα παραπάνω ένα Ν-γράμμα αποτελεί μια διανυσματική αναπαράσταση η οποία συμπεριλαμβάνει όλους τους συνδυασμούς Ν-γραμμάτων που εμφανίζονται στο αλφαριθμητικό. Ουσιαστικά κάθε διάνυσμα αποτελείται από μια διανυσματική συνιστώσα για κάθε πιθανό συνδυασμό Ν-γράμματος. Για παράδειγμα για ένα αλφάβητο που περιλαμβάνει τα γράμματα 'α' ως 'ω' και τους αριθμούς '0' ως '9' ο διανυσματικός χώρος θα είναι $34^N$ – διαστατός [10].

Η τεχνική των Ν-γραμμάτων ενδέχεται να εφαρμοσθεί με διάφορους τρόπους. Στην εργασία διερευνήθηκαν τρεις εναλλακτικές μέθοδοι χρήσης των Ν-γραμμάτων. Στη μια περίπτωση δημιουργείται ένα δυναμοσύνολο όλων των Ν-γραμμάτων που παράγονται από κάθε αλφαριθμητικό, και ελέγχεται ο αριθμός των κοινών Ν-γραμμάτων. Στην δεύτερη περίπτωση τροποποιείται η παραπάνω μέθοδος ώστε να μην

παράγεται το δυναμοσύνολο, αλλά μόνο ορισμένες υπο-λίστες Ν-γραμμάτων. Στην τρίτη περίπτωση γίνεται χρήση μόνο Τρι-γραμμάτων, και στο διάνυσμα με τα Τρι-γράμματα προστίθεται ένα ακόμη διάνυσμα, το οποίο περιλαμβάνει αριθμούς που αντιστοιχούν στο πόσες φορές εμφανίζεται το αντίστοιχο Τρι-γράμμα μέσα στο αλφαριθμητικό. Αυτές οι μέθοδοι θα επεξηγηθούν λεπτομερέστερα στη συνέχεια της εργασίας, στις ενότητες 3.2 και 3.3.

### 2.1.2 Τεχνική Χρήσης Απόστασης Levenshtein

Η απόσταση Levenshtein [35] είναι ένας αλγόριθμος ο οποίος εξετάζει την ομοιότητα μεταξύ δύο αλφαριθμητικών, και είναι ευρύτερα γνωστός με τον όρο Edit Distance [13], ο οποίος σε ελεύθερη μετάφραση σημαίνει απόσταση σύνταξης. Ουσιαστικά πήρε την συγκεκριμένη ονομασία από τον Vladimir Levenshtein ο οποίος υλοποίησε τον αλγόριθμο το 1965. Ο αλγόριθμος εξετάζει την ομοιότητα μεταξύ δύο αλφαριθμητικών, υπολογίζοντας τον ελάχιστο αριθμό πράξεων που χρειάζονται προκειμένου να επιτευχθεί η μετατροπή του ενός αλφαριθμητικού στο άλλο. Ο όρος πράξη αντιστοιχεί στην εισαγωγή, διαγραφή ή αντικατάσταση ενός μοναδικού χαρακτήρα στο αλφαριθμητικό. Η διαδικασία του υπολογισμού των πράξεων επιτυγχάνεται χρησιμοποιώντας έναν πίνακα που περιέχει τα δύο αλφαριθμητικά. Για παράδειγμα προκειμένου να υπολογιστεί η ομοιότητα μεταξύ των αλφαριθμητικών “Johanson” και “Johnsen” δημιουργείται ο παρακάτω πίνακας.

| | | **J** | **o** | **h** | **a** | **n** | **s** | **o** | **n** |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| **J** | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **o** | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| **h** | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| **n** | 4 | 3 | 2 | 1 | 1 | 1 | 2 | 3 | 4 |
| **s** | 5 | 4 | 3 | 2 | 2 | 2 | 1 | 2 | 3 |
| **e** | 6 | 5 | 4 | 3 | 3 | 3 | 2 | 2 | 3 |
| **n** | 7 | 6 | 5 | 4 | 4 | 3 | 3 | 3 | 2 |

Πίνακας 2-1: Πίνακας υπολογισμού απόστασης Levenshtein

Εν συνεχεία γίνεται η επεξήγηση της διαδικασίας υπολογισμού της απόστασης. Έστω ότι υφίστανται τα δύο αλφαριθμητικά $s_1$ και $s_2$. Το πλήθος των πράξεων $d$ υπολογίζεται συγκρίνοντας κάθε χαρακτήρα $l_1[i]$ του αλφαριθμητικού $s_1$ με κάθε χαρακτήρα $l_2[j]$ του αλφαριθμητικού $s_2$, ακολουθώντας τον παρακάτω αλγόριθμο:

$$Εάν\ l_1[i] = l_2[j]\ τότε\ \cos t = 0,\ αλλιώς\ \cos t = 1$$

$$\begin{aligned} d[i,j] = \min(&d[i-1,j]+1, && \leftarrow διαγραφή \\ &d[i,j-1]+1, && \leftarrow εισαγωγή \\ &d[i-1,j-1]+\cos t) && \leftarrow αντικατάσταση \end{aligned}$$

Όπως φαίνεται και στον πίνακα 2-1, ο συνολικός αριθμός των πράξεων που είναι απαραίτητες να γίνουν προκειμένου να μετατραπεί το ένα αλφαριθμητικό στο άλλο, περιέχεται στο κάτω – δεξιά στοιχείο του πίνακα.

Ένα ενδιαφέρον στοιχείο είναι η πολυπλοκότητα του αλγορίθμου. Εάν θεωρηθεί ότι το μήκος των δύο αλφαριθμητικών είναι $|s_1|$ και $|s_2|$, τότε η χρονική πολυπλοκότητα είναι $O(|s_1|*|s_2|)$. Δηλαδή εάν το μέγεθος των αλφαριθμητικών είναι περίπου ίσο με $n$, τότε η χρονική πολυπλοκότητα θα είναι $O(n^2)$. Η χωρική πολυπλοκότητα διαφέρει ανάλογα με τα δεδομένα που πρόκειται να αποθηκευτούν. Εάν αποθηκεύεται ολόκληρος ο πίνακας έτσι ώστε να είναι δυνατή η επανεξέταση του, η χωρική πολυπλοκότητα είναι $O(|s_1|*|s_2|)$, δηλαδή $O(n^2)$, ενώ εάν αποθηκεύεται μόνο η τιμή της απόστασης τότε η πολυπλοκότητα γίνεται $O(|s_1|)$, δηλαδή $O(n)$.

### 2.1.3 Τεχνική Χρήσης Απόστασης Jaro-Winkler

Η απόσταση Jaro-Winkler [46, 47] αποτελεί, όπως και η απόσταση Levenshtein, έναν αλγόριθμο ο οποίος συγκρίνει την ομοιότητα δύο αλφαριθμητικών εξετάζοντας το πλήθος των πράξεων εισαγωγής, διαγραφής και αντιμεταθέσεως χαρακτήρων, που χρειάζονται προκειμένου να γίνουν όμοια τα αλφαριθμητικά. Ο αλγόριθμος προτάθηκε από τον W. E. Winkler το 1999 και στην ουσία αποτελεί μια βελτιωμένη εκδοχή του αλγόριθμου που πρότεινε το 1989 ο M. A. Jaro [25].

Ο Jaro εισήγαγε έναν αλγόριθμο σύγκρισης αλφαριθμητικών ο οποίος υπολογίζει τιμές μερικής συμφωνίας μεταξύ των δύο αλφαριθμητικών. Ο συγκεκριμένος αλγόριθμος είναι ιδανικός στο να προσαρμόζει ακριβώς τα βάρη συμφωνίας μεταξύ δύο αλφαριθμητικών τα οποία δεν συμφωνούν σε μια βάση χαρακτήρα προς χαρακτήρα. Τα τρία βασικά συστατικά του αλγόριθμου Jaro είναι τα εξής:

1. Υπολογισμός του μήκους των αλφαριθμητικών.
2. Εύρεση του πλήθους των κοινών χαρακτήρων ανάμεσα στα δύο αλφαριθμητικά.
3. Εύρεση του αριθμού των αντιμεταθέσεων που πρέπει να γίνουν ώστε τα δύο αλφαριθμητικά να είναι όμοια.

Παίρνοντας υπόψιν τα παραπάνω δεδομένα, θεωρώντας ότι υπάρχουν δύο αλφαριθμητικά $s_1$ και $s_2$, ο αλγόριθμος του Jaro δίνεται από τον τύπο:

$$\text{jaro}(s_1, s_2) = \frac{1}{3}\left(\frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-\tau}{c}\right)$$

Όπου,

- $|s_1|$ είναι το μήκος του πρώτου αλφαριθμητικού,
- $|s_2|$ είναι το μήκος του δεύτερου αλφαριθμητικού,
- $\tau$ είναι το πλήθος των αντιμεταθέσεων χαρακτήρων που πρέπει να γίνουν,
- $c$ είναι το πλήθος των κοινών χαρακτήρων ανάμεσα στα δύο αλφαριθμητικά.

Στο σημείο αυτό είναι ιδιαίτερα σημαντικό να αναφερθεί ότι προκειμένου να θεωρηθούν κοινοί δύο χαρακτήρες από τα $s_1$ και $s_2$, η μεταξύ τους απόσταση οφείλει να είναι τουλάχιστον ίση με το μισό του μήκους του μικρότερου αλφαριθμητικού μειωμένο κατά μια μονάδα. Δηλαδή πρέπει να ισχύει ο τύπος:

$$H = \frac{\max\left(|s_1|, |s_2|\right)}{2} - 1$$

Οι κοινοί χαρακτήρες από τα δύο αλφαριθμητικά χαρακτηρίζονται ως εκχωρημένοι. Οι εναπομείναντες χαρακτήρες είναι μη εκχωρημένοι. Είναι φανερό ότι τα δύο αλφαριθμητικά θα έχουν τον ίδιο αριθμό εκχωρημένων χαρακτήρων.

Με βάση τους εκχωρημένους χαρακτήρες υπολογίζεται και το πλήθος των αντιμεταθέσεων. Ο πρώτος εκχωρημένος χαρακτήρας από το πρώτο αλφαριθμητικό συγκρίνεται με τον πρώτο εκχωρημένο χαρακτήρα από το δεύτερο αλφαριθμητικό. Κατόπιν συγκρίνονται οι δύο δεύτεροι εκχωρημένοι χαρακτήρες, και η ίδια διαδικασία ακολουθείται μέχρι να συγκριθούν και οι τελευταίοι. Εν τέλει το πλήθος των αντιμεταθέσεων υπολογίζεται από τον αριθμό των εκχωρημένων αλλά διαφορετικών χαρακτήρων, διαιρεμένο με το δύο.

Η βελτίωση που επέφερε ο Winkler στον αλγόριθμο του Jaro σχετίζεται με την πιθανή ύπαρξη ενός κοινού προθέματος ανάμεσα στα δύο αλφαριθμητικά. Θεωρώντας

λοιπόν την ύπαρξη δύο αλφαριθμητικών $s_1$ και $s_2$, η τελική έκδοση του αλγορίθμου Jaro – Winkler δίνεται από τον τύπο:

$$\text{Jaro}-\text{Winkler}(s_1,s_2)=\text{Jaro}(s_1,s_2)+i*0.1*(1-\text{Jaro}(s_1,s_2))$$

Όπου $i$ είναι ο αριθμός των χαρακτήρων που περιλαμβάνονται στο κοινό πρόθεμα.

Προκειμένου να γίνουν καλύτερα κατανοητά τα παραπάνω θα δοθεί ένα σύντομο παράδειγμα εφαρμογής του αλγόριθμου Jaro – Winkler. Έστω ότι υπάρχουν τα δύο αλφαριθμητικά "ALEKSANDER" και "ALEXANDRE". Η μεταξύ τους απόσταση υπολογίζεται ως εξής:

- Μέγιστη απόσταση κοινών χαρακτήρων $H = 4$.
- Πλήθος κοινών χαρακτήρων $c = 8$ (A,L,E,A,N,D,E,R).
- Πλήθος αντιμεταθέσεων $\tau = 1$.
- Η απόσταση Jaro είναι $\text{jaro}(s_1,s_2) = \frac{1}{3}\left(\frac{8}{10}+\frac{8}{9}+\frac{8-1}{8}\right) = 0.928125$ .
- Τελικά ισχύει jaro – winkler($s_1$,$s_2$) = 0.928125 + 3*0.1*(1 – 0.928125) = 0.9496875.

## 2.2 Τεχνικές Blocking

Η διασύνδεση εγγραφών ενδέχεται να έχει ιδιαίτερο κόστος όταν εφαρμόζεται σε πολύ μεγάλες Βάσεις Δεδομένων. Αυτό γίνεται εύκολα κατανοητό αν αναλογιστεί κανείς ότι προκειμένου να επιτευχθεί η όλη διαδικασία, πρέπει να συγκριθούν όλες οι εγγραφές της μιας Βάσης Δεδομένων με όλες τις εγγραφές της άλλης. Προκειμένου να μειωθεί το πλήθος των συγκρίσεων εφαρμόζεται μια διαδικασία που ονομάζεται Blocking [1, 3, 4, 28]. Στο Blocking κά θε Βάση Δεδομένων οργανώνει τις εγγραφές τις σε μπλοκ με βάση μια προκαθορισμένη τεχνική, και στη συνέχεια η διασύνδεση εφαρμόζεται μόνο μεταξύ εγγραφών που ανήκουν σε όμοια μπλοκ. Ουσιαστικά, σύμφωνα με τους Su Yan et al [49], «ένα μπλοκ είναι μια συλλογή από εγγραφές οι οποίες ενδεχομένως να

αποτελούν διπλότυπα η μια της άλλης». Η τεχνικές που παρουσιάζονται στη συνέχεια είναι το απλό Blocking [8], η ταξινομημένη γειτνίαση [8], το Blocking βασισμένο σε πίνακα επιθεμάτων [8], και το Blocking βασισμένο στην χρήση μετρικών απόστασης [8].

### 2.2.1 Απλό Blocking

Το απλό Blocking, το οποίο είναι επίσης γνωστό ως τυποποιημένο Blocking, είναι μια διαδικασία που εφαρμόζεται στον τομέα της διασύνδεσης εγγραφών για αρκετές δεκαετίες [17]. Σύμφωνα με αυτή την τεχνική οι εγγραφές οργανώνονται σε μπλοκ με βάση ένα συγκεκριμένο κλειδί Blocking, και η σύγκριση λαμβάνει χώρα μόνο ανάμεσα σε εγγραφές που ανήκουν στο ίδιο μπλοκ. Επίσης κάθε εγγραφή μπορεί να ανήκει μόνο σε ένα μπλοκ. Ως κλειδί είναι δυνατόν να οριστεί ένα πεδίο των εγγραφών, ή ένα πρόθεμα από ένα πεδίο των εγγραφών, ή ακόμη μια αλληλουχία προθεμάτων από δύο ή περισσότερα πεδία των εγγραφών. Παραδείγματος χάριν ως κλειδί μπορεί να οριστούν οι τέσσερις πρώτοι χαρακτήρες του πεδίου «Επώνυμο», ή η αλληλουχία των χαρακτήρων αυτών με τους τρεις πρώτους χαρακτήρες του πεδίου «Αριθμός Ασφάλισης».

Είναι φανερό ότι στο απλό Blocking το μέγεθος των μπλοκ που συνθέτονται εξαρτάται από το είδος του κλειδιού που επιλέγεται, και για αυτό είναι ιδιαίτερα δύσκολο να προβλεφθεί ο συνολικός αριθμός των υποψήφιων προς σύγκριση εγγραφών. Για παράδειγμα αν το κλειδί είναι το πεδίο «Φύλο» τότε θα δημιουργηθούν συνολικά δυο μπλοκ. Από την άλλη αν το κλειδί είναι το πεδίο «Αριθμός Ασφάλισης» το πιο πιθανό είναι κάθε εγγραφή να συνιστά και διαφορετικό μπλοκ. Ενδεικτικά αναφέρεται ότι για δύο σύνολα δεδομένων με $n$ εγγραφές το καθένα, όπου δημιουργούνται $b$ ισομεγέθη μπλοκ με $n/b$ εγγραφές το καθένα, το πλήθος των συγκρίσεων θα είναι $O(n^2/b)$.

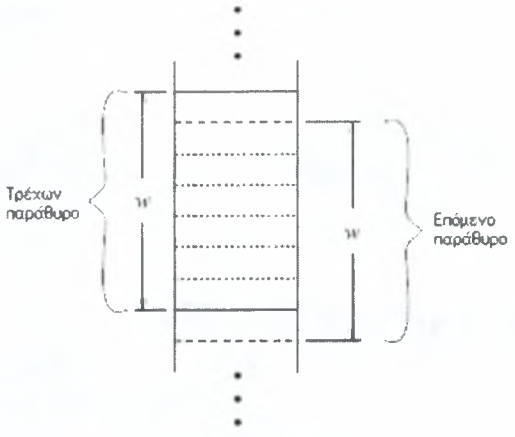
Το βασικό πρόβλημα που παρουσιάζει αυτή η τεχνική είναι ότι αν μια εγγραφή έχει κάποιο λάθος στους χαρακτήρες που καθορίζουν το κλειδί, τότε θα τοποθετηθεί σε

λάθος μπλοκ. Το πρόβλημα αυτό μπορεί να λυθεί, με κάποιο κόστος σε υπολογιστικούς πόρους, αν καθοριστούν περισσότερα του ενός κλειδιά από διαφορετικά πεδία το καθένα. Κατόπιν θα γίνει η διασύνδεση επαναληπτικά για κάθε κλειδί και η ένωση των επιμέρους αποτελεσμάτων θα αποτελεί το τελικό αποτέλεσμα της σύγκρισης.

### 2.2.2 Ταξινομημένη Γειτνίαση

Η ταξινομημένη γειτνίαση αποτελεί μια βελτιωμένη εκδοχή του απλού Blocking. Η βασική ιδέα της μεθόδου είναι η ταξινόμηση των διαφορετικών τιμών που έχουν οι εγγραφές για κλειδί Blocking, και ο καθορισμός ενός παραθύρου μεγέθους $w$ το οποίο θα μετακινείται σειριακά πάνω στις ταξινομημένες τιμές, όπως διαφαίνεται στο σχήμα 2-1. Ένα μπλοκ καθορίζεται από όλες τις τιμές κλειδιού Blocking που βρίσκονται μέσα στο παράθυρο. Όπως γίνεται κατανοητό όταν το μέγεθος του παραθύρου είναι μεγαλύτερο της μονάδος, $w > 1$, τότε στο ίδιο μπλοκ δεν θα ανήκουν μόνο οι εγγραφές που έχουν την ίδια τιμή κλειδιού, αλλά και εγγραφές με παραπλήσιες τιμές. Αντίθετα, εάν $w = 1$ τότε η τεχνική της ταξινομημένης γειτνίασης είναι όμοια με την τεχνική του απλού Blocking.

Ένα ενδιαφέρον στοιχείο που παρατηρείται είναι ότι για όλα τα παράθυρα που είναι μεγαλύτερα της μονάδος, τα υποψήφια προς σύγκριση ζεύγη εγγραφών που παράγονται, θα αποτελούν υπερσύνολο των ζευγών που παράγονται από το απλό Blocking. Και γενικότερα, για δύο διαφορετικά μεγέθη παραθύρου $w_i$ και $w_j$, όπου ισχύει $w_i < w_j$, όλα τα ζεύγη εγγραφών που παράγονται στο παράθυρο με μέγεθος $w_i$ θα ανήκουν και στα ζεύγη που παράγονται στο παράθυρο με μέγεθος $w_j$ [8].




Σχήμα 2-1: Διάγραμμα της Ταξινομημένης Γειτνίασης

Όταν η συγκεκριμένη τεχνική εφαρμόζεται σε *n* εγγραφές, τότε το παράθυρο μετακινείται συνολικά $n - w + 1$ φορές, και το πλήθος των μπλοκ που παράγονται είναι πάλι $n - w + 1$ [49]. Επίσης οι πιθανές συγκρίσεις για κάθε εγγραφή είναι $2w - 1$ και ο συνολικός αριθμός των συγκρίσεων που πρόκειται να γίνουν είναι $O(nw)$.

Όπως και στο απλό Blocking, το βασικό πρόβλημα της ταξινομημένης γειτνίασης είναι ότι αν υπάρχουν λάθη στην αρχή κάποιας τιμής κλειδιού, τότε παρόμοιες εγγραφές δεν θα ανήκουν στο ίδιο μπλοκ και δεν θα επιτευχθεί η σύγκριση τους. Αυτό γίνεται φανερό αν δύο τιμές κλειδιού είναι το «Cate» και «Kate», οπότε κατά την ταξινόμηση οι δύο εγγραφές θα είναι αρκετά μακριά ώστε να συγκριθούν.

### 2.2.3 Blocking με Πίνακες Επιθεμάτων

Η χρήση πινάκων επιθεμάτων στο Blocking είναι μια σχετικά πρόσφατα ανεπτυγμένη τεχνική, η οποία βασίζεται στην δημιουργία πινάκων οι οποίοι περιέχουν τις τιμές του κλειδιού Blocking και τα επιθέματα τους. Στην ουσία ένας πίνακας επιθεμάτων περιέχει αλφαριθμητικά ή ακολουθίες χαρακτήρων και τα επιθέματα τους σε ταξινομημένη σειρά. Στην προκειμένη περίπτωση στον πίνακα επιθεμάτων εισάγονται μόνο επιθέματα που έχουν ένα προκαθορισμένο ελάχιστο μήκος *min_len,* και τα υπόλοιπα απορρίπτονται. Παραδείγματος χάριν στην περίπτωση που το κλειδί Blocking

είναι το αλφαριθμητικό "johnson" και το προκαθορισμένο μήκος είναι *min_len = 3*, τότε στον πίνακα επιθεμάτων εισάγονται οι τιμές "johnson", "ohnson", "hnson", "nson" και "son". Κάθε μια από τις παραπάνω τιμές αποτελεί και ένα ξεχωριστό μπλοκ, και οι εγγραφές που περιέχουν αυτά τα αλφαριθμητικά στο πεδίο του κλειδιού Blocking εισάγονται στα αντίστοιχα μπλοκ. Δηλαδή μια εγγραφή ανάλογα με την τιμή που έχει για κλειδί Blocking, είναι δυνατόν να ανήκει σε περισσότερα του ενός μπλοκ. Συγκεκριμένα αν έχει για τιμή ένα αλφαριθμητικό που περιέχει *c* χαρακτήρες, τότε θα εισαχθεί σε $(c - min\_len + 1)$ μπλοκ [8].

Η τεχνική χρήσης πινάκων επιθεμάτων παρέχει την δυνατότητα να παρακαμφθεί το πρόβλημα της δημιουργίας μεγάλων μπλοκ εγγραφών, το οποίο συνεπάγεται μεγάλο αριθμό συγκρίσεων και κατανάλωση πολλών υπολογιστικών πόρων. Αυτό επιτυγχάνεται με τον ορισμό ενός αριθμού μεγίστου μεγέθους μπλοκ, *max_block*. Τα μπλοκ που περιέχουν λιγότερες εγγραφές από το *max_block* οδηγούνται προς σύγκριση, ενώ τα μπλοκ με περισσότερες εγγραφές απορρίπτονται. Για παράδειγμα αν ισχύει *max_block = 20,* και η τιμή του πίνακα επιθεμάτων "son" περιέχει 50 εγγραφές, τότε το "son" θεωρείται πολύ γενικό και οι εγγραφές του δεν συγκρίνονται. Αντίθετα αν το "hnson" αποτελείται από 18 εγγραφές τότε αυτές θα οδηγηθούν στη σύγκριση.

### 2.2.4 Blocking με Χρήση Μετρικών Απόστασης

Η τεχνική Blocking με χρήση μετρικών απόστασης βασίζεται στην απεικόνιση των τιμών των κλειδιών Blocking σε μια Ευκλείδεια απόσταση. Ουσιαστικά οι τιμές αυτές είναι κάποια αλφαριθμητικά τα οποία απεικονίζονται σε αντικείμενα της Ευκλείδειας απόστασης. Η απεικόνιση μπορεί να πραγματοποιηθεί χρησιμοποιώντας μια τεχνική εφαρμογής μετρικών απόστασης, όπως είναι η απόσταση Levenshtein ή η απόσταση Jaro – Winkler, που περιγράφονται στις ενότητες 2.1.2 και 2.1.3 αντίστοιχα. Με βάση αυτές τις αποστάσεις, αν δυο τιμές κλειδιού απέχουν λιγότερο από μια προκαθορισμένη μέγιστη απόσταση *max_dist,* τότε οι εγγραφές τους κατηγοριοποιούνται στο ίδιο μπλοκ. Παραδείγματος χάριν, χρησιμοποιώντας την τεχνική Jaro – Winkler, αν έχει

οριστεί *max_dist = 0.85,* τότε τα αλφαριθμητικά "johnson" και "johnsen" που απέχουν *0.94* θα εισαχθούν στο ίδιο μπλοκ. Αντίθετα τα αλφαριθμητικά "miller" και "moler" που απέχουν *0.84* δεν θα κατηγοριοποιηθούν στο ίδιο μπλοκ.

Η χρήση της συγκεκριμένης τεχνικής παρέχει την δυνατότητα στον χρήστη να μεταβάλει αρκετά εύκολα το μέγεθος των μπλοκ, ώστε να πετύχει την βέλτιστη λύση ανάμεσα στην κατανάλωση υπολογιστικών πόρων και επιτυχίας στο ταίριασμα των εγγραφών. Ο χρήστης έχει την δυνατότητα να μεγαλώνει ή να μειώνει το μέγεθος των μπλοκ ανάλογα με τις απαιτήσεις του, αυξομειώνοντας την τιμή της προκαθορισμένης απόστασης *max_dist.*

## 2.3 Κρυπτογραφικοί Ορισμοί

Σε αυτήν την ενότητα θα γίνει η επεξήγηση των κρυπτογραφικών τεχνικών που εφαρμόσθηκαν στην εργασία. Αρχικά κρίνεται σκόπιμο να δοθεί ένας ορισμός στον όρο κρυπτογραφία. Με τον όρο κρυπτογραφία [30] εννοούμε τον επιστημονικό κλάδο ο οποίος ασχολείται με την μελέτη, τη χρήση και την ανάπτυξη τεχνικών κρυπτογράφησης και αποκρυπτογράφησης για την απόκρυψη των περιεχομένων των μηνυμάτων ή των δεδομένων, και την διευκόλυνση της ανίχνευσης κακόβουλων μετατροπών στα μηνύματα.

Στην εργασία εφαρμόσθηκαν δύο διαφορετικές τεχνικές. Η μια είναι η χρήση του κρυπτογραφικού αλγόριθμου Twofish [37, 53] και η άλλη είναι η εφαρμογή της συνάρτησης κατακερματισμού MD5 [54, 59].

### 2.3.1 Κρυπτογραφικός Αλγόριθμος TwoFish

Ο TwoFish αποτελεί έναν κρυπτογραφικό αλγόριθμο δέσμης που για την κρυπτογράφηση χρησιμοποιεί ένα συμμετρικό (μυστικό) κλειδί. Ο συγκεκριμένος αλγόριθμος σχεδιάστηκε στηριζόμενος στον κρυπτογραφικό αλγόριθμο AES [16, 58], ο οποίος αποτελεί το πρότυπο κρυπτογραφικού αλγορίθμου δέσμης. Ο AES, και κατά συνέπεια ο TwoFish, είναι ιδιαίτερα ανθεκτικός στις επιθέσεις και παρέχει υψηλότατο

επίπεδο ασφάλειας [30, 37, 53], και για αυτόν τον λόγο επιλέχθηκε να χρησιμοποιηθεί. Συνοπτικά τα βασικά χαρακτηριστικά του αλγορίθμου TwoFish είναι τα εξής:

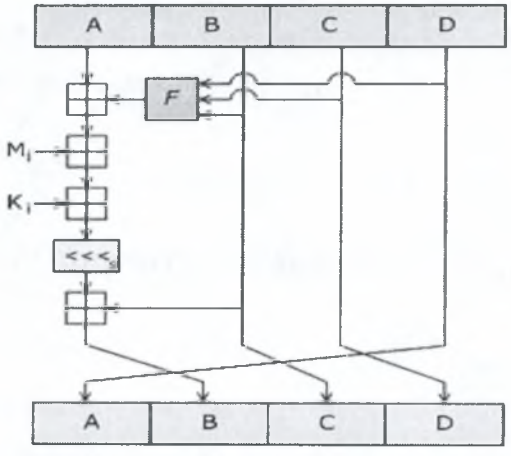
- Είναι συμμετρικός αλγόριθμος δέσμης, με μέγεθος για κάθε μπλοκ 128 bits.
- Το κλειδί που χρησιμοποιεί μπορεί να έχει μέγεθος 128 bits, 192 bits και 256 bits.
- Σε περίπτωση που ο χρήστης που επιλέγει το μέγεθος του κλειδιού επιλέξει κάποιον άλλον αριθμό, τότε ο αλγόριθμος μόνος του το προσαρμόζει στο κοντινότερο δυνατό μέγεθος.
- Δεν χρησιμοποιούνται ποτέ αδύναμα κλειδιά.
- Η χρήση του υποστηρίζεται από πληθώρα λογισμικών προγραμμάτων, από διαφορετικά λειτουργικά συστήματα, καθώς και από διάφορα είδη τεχνολογίας επεξεργαστών.

### 2.3.2 Συνάρτηση Κατακερματισμού MD5

Η μαθηματική συνάρτηση μονόδρομου κατακερματισμού MD5 χρησιμοποιήθηκε προκειμένου να επιτευχθεί υψηλότερο επίπεδο ασφάλειας των πληροφοριών. Η συνάρτηση, η οποία βασίζεται στο κρυπτογραφικό σύστημα RSA, δέχεται σαν είσοδο ένα μήνυμα οποιουδήποτε μεγέθους, και στην συνέχεια παράγει σαν έξοδο μια κατακερματισμένη εκδοχή του αρχικού μηνύματος, η οποία έχει μέγεθος 128 bits. Ουσιαστικά με τον όρο κατακερματισμένη, εννοείται ότι παράγεται μια σύνοψη του αρχικού μηνύματος που θα είναι πάντα 128 bits όσο μέγεθος και αν έχει το αρχικό μήνυμα.

Ο λόγος που χρησιμοποιείται η συγκεκριμένη συνάρτηση είναι για να μην έχει την δυνατότητα κάποιος τρίτος να παρέμβει στα ήδη κρυπτογραφημένα δεδομένα και να τα παραποιήσει. Αυτό βασίζεται στην έννοια της μονόδρομης συνάρτησης. Ουσιαστικά υπάρχει μια μαθηματική συνάρτηση, η οποία εφαρμόζεται στο μήνυμα εισόδου και παράγει την σύνοψη του. Όμως δεν είναι δυνατόν να υπάρξει μια συνάρτηση που να εφαρμόζει την ακριβώς αντίστροφη διαδικασία, δηλαδή να δέχεται σαν είσοδο το κατακερματισμένο μήνυμα και να το επαναφέρει στην αρχική εκδοχή του. Επομένως η

συγκεκριμένη διαδικασία κρίνεται αρκετά ασφαλής ως προς την μετάδοση μη παραποιημένων δεδομένων. Όπως θα εξηγηθεί στην ενότητα 3.4, όποτε κρίνεται σκόπιμο είναι δυνατόν να εφαρμοστεί στα δεδομένα μόνο η συνάρτηση κατακερματισμού χωρίς την χρήση κάποιου κρυπτογραφικού αλγορίθμου. Αυτό δεν σημαίνει ότι θα υπάρξει έλλειμμα ασφάλειας των δεδομένων, αφού δεν μπορεί κάποιος να επαναφέρει στην αρχική τους μορφή τα κατακερματισμένα δεδομένα. Στο παρακάτω σχήμα παρουσιάζεται μια διαδικασία της συνάρτησης του MD5:




Σχήμα 2-2: Μια Διαδικασία του αλγορίθμου MD5

Ο αλγόριθμος MD5 αποτελείται από 64 τέτοιες διαδικασίες, οι οποίες είναι οργανωμένες σε 4 γύρους των 16 διαδικασιών. Το F αποτελεί μια μη γραμμική συνάρτηση. Σε κάθε γύρο χρησιμοποιείται μια μόνο τέτοια συνάρτηση. Το $M_i$ δηλώνει μια δέσμη μεγέθους 32 bits της αρχικής έκδοσης και το $K_i$ μια σταθερά μεγέθους κι αυτή 32 bits, η οποία είναι διαφορετική για κάθε διαδικασία. Το σύμβολο <<<s δηλώνει μια μετατόπιση των bit προς τα αριστερά κατά 's' θέσεις, και μπορεί να παίρνει διαφορετικές τιμές ανάλογα με την διαδικασία. Τέλος το σύμβολο ⊞ δηλώνει την πρόσθεση που γίνεται κάθε φορά, πάντα σε modulo $2^{32}$.

# Κεφάλαιο 3

# Πρωτόκολλα Ασφαλούς Διασύνδεσης Γνωρισμάτων

Σε αυτό το κεφάλαιο θα γίνει η παρουσίαση των πρωτοκόλλων που υλοποιήθηκαν στα πλαίσια της έρευνας για την ασφαλή διασύνδεση των εγγραφών κατανεμημένων Βάσεων Δεδομένων. Σε αυτήν την κατεύθυνση θα περιγραφθεί αρχικά το ευρύτερο πλαίσιο πάνω στο οποίο βασίστηκαν όλα τα πρωτόκολλα. Κατόπιν θα γίνει η διατύπωση των πρωτοκόλλων που επιτυγχάνουν την σύγκριση της ομοιότητας των γνωρισμάτων με χρήση Ν-γραμμάτων και δυναμοσυνόλων, με χρήση Τρι-γραμμάτων, και με την χρήση των μετρικών απόστασης Levenshtein και Jaro – Winkler. Εκτός των άλλων θα δοθεί και η εναλλακτική μορφή των πρωτοκόλλων που περιλαμβάνει την οργάνωση των εγγραφών σε μπλοκ, προτού διεκπεραιωθεί η σύγκριση τους.

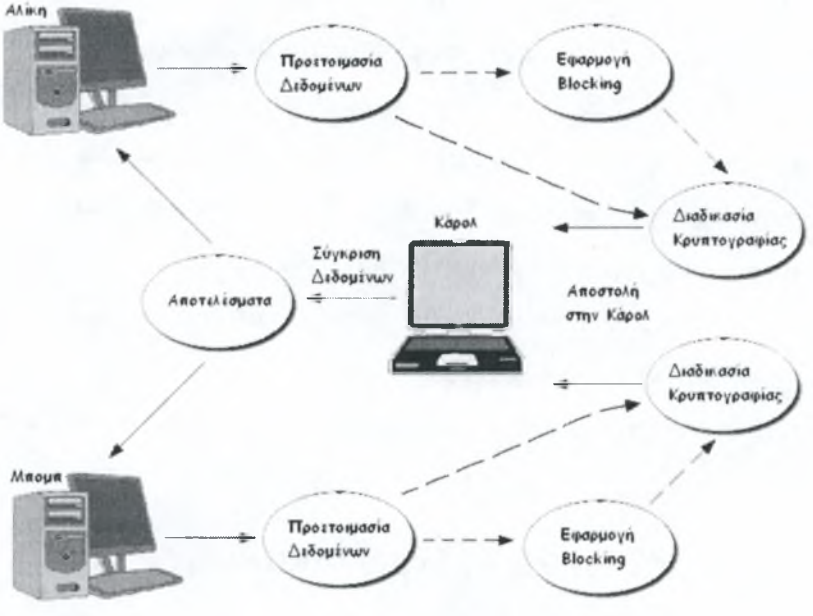
## 3.1 Γενικό Πλαίσιο Πρωτοκόλλων

Το πλαίσιο μέσα στο οποίο κινήθηκαν τα πρωτόκολλα βασίζεται στο μοντέλο της τρίτης έμπιστης πηγής [20]. Θεωρείται ότι υπάρχουν δύο Βάσεις Δεδομένων οι οποίες πρόκειται να προχωρήσουν στην σύγκριση της ομοιότητας των πληροφοριών τους, με την βοήθεια μιας τρίτης έμπιστης πηγής η οποία εγγυάται την διατήρηση της ασφάλειας.

Για χάριν απλούστευσης, έστω ότι υπάρχει η Αλίκη η οποία έχει στην κατοχή της μια Βάση Δεδομένων Α, και ο Μπομπ ο οποίος έχει στην κατοχή του μια Βάση Δεδομένων Β. Η Α περιέχει μια ή περισσότερες εγγραφές $r_1$, $r_2$ κ.ο.κ., κάθε μια από τις οποίες έχει κάποιον αριθμό από πεδία $r_{1.1}$, $r_{2.2}$ κ.ο.κ., που συμπεριλαμβάνουν

ευαίσθητα αλφαριθμητικά όπως είναι ονόματα και διευθύνσεις. Αντίστοιχα η B περιέχει μια ή περισσότερες εγγραφές $rec_1$, $rec_2$ κ.ο.κ., κάθε μια από τις οποίες έχει κάποιον αριθμό από πεδία $rec_{1.1}$, $rec_{2.2}$. Η ανάγκη για ασφάλεια οφείλεται στο γεγονός ότι τα περιεχόμενα των πεδίων ενδέχεται να οδηγούν στην αναγνώριση προσώπων ή στην αποκάλυψη εμπορικών ανταγωνιστικών πληροφοριών.

Η γενικότερη διαδικασία που ακολουθείται και από τα τέσσερα πρωτόκολλα, ανάλογα αν εφαρμόζουν Blocking ή όχι φαίνεται στο σχήμα 3-1.




Σχήμα 3-1: Αναπαράσταση Διαδικασίας Πρωτοκόλλων

Ο στόχος της Αλίκης και του Μπομπ είναι να προσδιορίσουν κατά πόσο οι πληροφορίες που περιέχονται στα $r_{i.i}$ ταιριάζουν με αυτές των $rec_{j.j}$, χωρίς να αποκαλύπτουν ο ένας στον άλλον ποιες είναι οι πραγματικές πληροφορίες. Για τον λόγο αυτό υπάρχει η τρίτη έμπιστη πηγή, η οποία θα ονομάζεται Κάρολ. Θεωρείται δεδομένο ότι η Κάρολ είναι υπεύθυνη ώστε:

- Να ακολουθήσει επακριβώς το κάθε πρωτόκολλο.
- Να μην αποκαλύψει πληροφορίες σε άλλες πηγές εκτός από όπου επιτρέπεται με βάση το εκάστοτε πρωτόκολλο.

- Να μην προσπαθήσει να προσδιορίσει τις πηγαίες πληροφορίες της Αλίκης και του Μπομπ.

Σε καμία περίπτωση δεν θεωρείται δεδομένο ότι η Αλίκη και ο Μπομπ εμπιστεύονται ο ένας τον άλλον. Τέλος πρέπει να αναφερθεί ότι για όλες τις μεταφορές δεδομένων μεταξύ των τριών μελών έχει συμφωνηθεί να χρησιμοποιηθεί κρυπτογράφηση συμμετρικού κλειδιού, ώστε να επιτυγχάνεται η ασφάλεια των δεδομένων και η αυθεντικοποίηση των μελών.

## 3.2 Σύγκριση Ομοιότητας με Χρήση N-γραμμάτων με Δυναμοσύνολα

Το πρώτο πρωτόκολλο που υλοποιήθηκε κατά την διάρκεια της έρευνας χρησιμοποιεί N-γράμματα και δυναμοσύνολα προκειμένου να επιτύχει την διασύνδεση των γνωρισμάτων. Το συγκεκριμένο πρωτόκολλο έχει προταθεί από τους T. Churches et al [10]. Εν συνεχεία παρουσιάζονται τα βήματα που ακολουθούνται κατά την εκτέλεση του πρωτοκόλλου, με βάση το μοντέλο της τρίτης έμπιστης πηγής που προτάθηκε στην ενότητα 3.1:

1. Αρχικά η Αλίκη και ο Μπομπ αποφασίζουν από κοινού για το ποιο θα είναι το μυστικό κλειδί κρυπτογράφησης, το οποίο θα γνωρίζουν μόνον αυτοί οι δυο, και καθορίζουν ποιον συγκεκριμένο αλγόριθμο κρυπτογράφησης θα εφαρμόσουν στα δεδομένα τους. Επίσης ορίζουν την μονόδρομη μαθηματική συνάρτηση κατακερματισμού, όπως είναι η MD5, που πρόκειται να χρησιμοποιήσουν. Παράλληλα συμφωνούν να εφαρμόσουν στα δεδομένα τους ένα τυποποιημένο πρωτόκολλο καθαρισμού των αλφαριθμητικών, ώστε οι πληροφορίες που περιέχονται στις εγγραφές τους να είναι στην ίδια τυποποιημένη μορφή.
2. Έπειτα η Αλίκη υπολογίζει μια ταξινομημένη λίστα με N-γράμματα για κάθε αλφαριθμητικό που περιέχεται στα πεδία $r_{i,i}$. Αξίζει να σημειωθεί ότι τα διπλότυπα N-γράμματα απαλείφονται, οπότε στην λίστα κάθε ένα εμφανίζεται

μια και μοναδική φορά. Κατόπιν η Αλίκη υπολογίζει όλες τις πιθανές υπό-λίστες με μήκος μεγαλύτερο το μηδενός, για κάθε λίστα Ν-γραμμάτων. Ουσιαστικά υπολογίζει το δυναμό-σύνολο των Ν-γραμμάτων αφαιρώντας το κενό σύνολο. Για παράδειγμα αν ένα πεδίο περιείχε το αλφαριθμητικό 'Peter', και ήθελε να υπολογίσει τα δι-γράμματα τότε θα παραγόταν η λίστα δι-γραμμάτων ("er", "et", "pe", "te"), και το δυναμό-σύνολο:

("er"), ("et"), ("pe"), ("te"),

("er", "et"), ("er", "pe"), ("er", "te"), ("et", "pe"), ("et", "te"), ("pe", "te"),

("er", "et", "pe"), ("er", "et", "te"), ("er", "pe", "te"), ("et", "pe", "te"),

("er", "et", "pe", "te")

Παρατηρούμε ότι αφού η λίστα περιέχει 4 δι-γράμματα τότε το μήκος των υπό-λιστών δι-γραμμάτων που υπολογίζονται κυμαίνεται από 1 έως 4. Εκτός των άλλων, συμπεραίνουμε ότι αν μια λίστα Ν-γραμμάτων περιέχει 'a' Ν-γράμματα, τότε ο συνολικός αριθμός των υπό-λιστών που θα απαρτίζουν το δυναμό-σύνολο θα είναι $2^a - 1$. Έπειτα η Αλίκη κρυπτογραφεί κάθε μια από τις υπολογισμένες υπό-λίστες Ν-γραμμάτων, χρησιμοποιώντας το προκαθορισμένο κλειδί και αλγόριθμο, και συνοψίζει το αποτέλεσμα εφαρμόζοντας την μονόδρομη συνάρτηση κατακερματισμού. Παράλληλα δημιουργεί μια κρυπτογραφημένη έκδοση του κλειδιού αναγνώρισης κάθε εγγραφής. Στη συνέχεια στέλνει στην Κάρολ αυτά που υπολόγισε προηγουμένως.

3. Ομοίως με την Αλίκη, ο Μπομπ φέρνει σε πέρας την ίδια διαδικασία που περιγράφηκε στο βήμα 2, για τις δικές του εγγραφές $rec_j$, και στέλνει τα αποτελέσματα στην Κάρολ.
4. Τέλος η Κάρολ, αφού έχει στην διάθεση της τα αποτελέσματα και των δυο, καθορίζει την τομή των συνόλων των τιμών των κρυπτογραφημένων Ν-γραμμάτων που της έχουν αποσταλεί. Κατόπιν για κάθε τιμή των Ν-γραμμάτων που ανήκουν στην τομή, για κάθε μοναδικό ζευγάρι των κλειδιών αναγνώρισης των εγγραφών, υπολογίζει την βαθμολογία σύγκρισης Ν-γραμμάτων. Η βαθμολογία δίνεται από τον τύπο:

$$\text{βαθμολογία} = \frac{\text{αριθμός\_κοινών\_N - γραμμάτων}}{0.5 * (\text{αριθμός\_N - γραμμάτων\_στο\_r}_{1,1} + \text{αριθμός\_N - γραμμάτων\_στο\_rec}_{1,1})}$$

Στη συνέχεια για κάθε μοναδικό ζευγάρι κλειδιών αναγνώρισης εγγραφών, από όλες τις βαθμολογίες που έχουν υπολογιστεί επιλέγει την μέγιστη. Είναι προφανές ότι οι τιμές της βαθμολογίας κυμαίνονται από 0 ως 1, με 1 να σημαίνει ότι οι κρυπτογραφημένες τιμές ταιριάζουν επακριβώς.

Αφού έχουν ολοκληρωθεί και τα τέσσερα βήματα του πρωτοκόλλου, ουσιαστικά έχουν παραχθεί τα αποτελέσματα της σύγκρισης. Τότε η Κάρολ θα παραδώσει τις βαθμολογίες μαζί με τους κωδικούς αναγνώρισης εγγραφών στην Αλίκη και τον Μπομπ, οι οποίοι στην συνέχεια θα αποφασίσουν σε ποιες ενέργειες θα προβούν. Ενδεχομένως, αντί να τους δώσει όλα τα αποτελέσματα, μπορεί να έχει οριστεί από πριν ένα κατώφλι βαθμολογίας, και να τους παραθέτει όλες τις βαθμολογίες που το ξεπερνούν, ώστε να αποφεύγεται η μετάδοση μεγάλου όγκου πληροφοριών. Αξίζει να τονισθεί ότι λόγω της κρυπτογράφησης, ο όγκος των δεδομένων που μεταδίδονται καθόλη την διάρκεια της διαδικασίας είναι πολύ μεγαλύτερος από ότι θα ήταν.

Όπως προκύπτει από τα πειραματικά αποτελέσματα τα οποία παρουσιάζονται στο κεφάλαιο 5, το συγκεκριμένο πρωτόκολλο αντιμετωπίζει ιδιαίτερες δυσκολίες στην αντιμετώπιση μεγάλων συνόλων δεδομένων λόγω της παραγωγής υπερβολικά μεγάλου αριθμού N-γραμμάτων. Για τον λόγο αυτό υλοποιήθηκε μια τροποποιημένη έκδοση του παραπάνω πρωτοκόλλου η οποία αντιμετωπίζει με επιτυχία τις δυσκολίες. Στην συγκεκριμένη έκδοση προκειμένου να αντιμετωπιστεί το πρόβλημα της παραγωγής πολλών N-γραμμάτων, για κάθε αλφαριθμητικό δεν υπολογίζεται ολόκληρο το δυναμοσύνολο N-γραμμάτων αλλά μόνον οι δύο υπο-λίστες που έχουν μήκος ίσο με ένα και δύο. Για να γίνει αυτό κατανοητό, στο παράδειγμα που προτάθηκε προηγουμένως με το αλφαριθμητικό “Peter”, όπου παράγεται η λίστα δι-γραμμάτων (“er”, “et”, “pe”, “te”), γίνεται ο υπολογισμός μόνο των υπο-λιστών:

(“er”), (“et”), (“pe”), (“te”) και

(“er”, “et”), (“er”, “pe”), (“er”, “te”), (“et”, “pe”), (“et”, “te”), (“pe”, “te”).

Επίσης θεωρήθηκε ότι η σύγκριση της ομοιότητας των εγγραφών μπορεί να επιτευχθεί σωστά εφαρμόζοντας την μόνο σε έναν συγκεκριμένο αριθμό πεδίων και όχι σε όλα, αρκεί να επιλεχθούν τα κατάλληλα πεδία. Οπότε στην τροποποιημένη εκδοχή του πρωτοκόλλου επιλέγονται δύο πεδία προς σύγκριση. Είναι προφανές ότι πλέον θα παράγονται N-γράμματα μόνο για τις τιμές που περιλαμβάνονται σε αυτά τα πεδία και όχι στα υπόλοιπα. Δύο πεδία που είναι κατάλληλα για την σύγκριση είναι το «επώνυμο» και το «όνομα». Παράλληλα είναι δυνατόν να επιλεχθεί να εφαρμοσθεί το πρωτόκολλο περισσότερες της μιας φορές, εφαρμόζοντας την σύγκριση σε διαφορετικά πεδία κάθε φορά. Αυτό πιθανόν να δώσει καλύτερα αποτελέσματα αλλά συνεπάγεται κατανάλωση περισσότερων υπολογιστικών πόρων.

Επομένως, σύμφωνα με το τροποποιημένο πρωτόκολλο, στο πρώτο βήμα της διαδικασίας, εκτός των άλλων, η Αλίκη και ο Μπομπ θα συνεννοούνται για τα πεδία στα οποία θα εφαρμόσουν την σύγκριση. Κατόπιν στο δεύτερο και τρίτο βήμα θα υπολογίζουν μόνο της δύο υπο-λίστες ταξινομημένων N-γραμμάτων, και στο τέταρτο βήμα η Κάρολ θα εφαρμόζει την σύγκριση μόνο για τα N-γράμματα που υπολογίστηκαν από τα δύο πεδία.

## 3.3 Σύγκριση Ομοιότητας με Χρήση Τρι-γραμμάτων

Το δεύτερο πρωτόκολλο που υλοποιήθηκε κατά την διάρκεια της εργασίας επιτυγχάνει την διασύνδεση των γνωρισμάτων κάνοντας χρήση μόνο Τρι-γραμμάτων. Στην ουσία αποτελεί συνδυασμό του μοντέλου ασφάλειας της τρίτης έμπιστης πηγής και της μεθόδου σύγκρισης αλφαριθμητικών που έχει προταθεί από τον J. Hylton [23]. Παρακάτω γίνεται η επεξήγηση των βημάτων του πρωτοκόλλου:

1. Ομοίως με το πρωτόκολλο της ενότητας 3.2, η Αλίκη και ο Μπομπ αποφασίζουν από κοινού για το μυστικό κλειδί κρυπτογράφησης, καθορίζουν τον αλγόριθμο κρυπτογράφησης που θα εφαρμόσουν, ορίζουν την μονόδρομη μαθηματική συνάρτηση κατακερματισμού, και φέρουν τα δεδομένα τους στην ίδια τυποποιημένη μορφή.

2. Έπειτα η Αλίκη υπολογίζει ένα διάνυσμα Τρι-γραμμάτων $\vec{A}$ για κάθε αλφαριθμητικό *s* που περιέχεται στα πεδία $r_{i.i}$. Το συγκεκριμένο διάνυσμα καθορίζεται ως εξής:

$$\vec{A} = \{a_{aaa}, a_{aab}, \ldots, a_{d5f}, \ldots, a_{999}\}$$

Όπου $a_{aaa}$ είναι το πλήθος των εμφανίσεων του "*aaa*" στο αλφαριθμητικό *s*.

Για να γίνει αυτό κατανοητό θα τεθεί το παρακάτω παράδειγμα. Ως γνωστών το αλφαριθμητικό "record" αποτελείται από τα Τρι-γράμματα "rec", "eco", "cor", και "ord". Εάν το Τρι-γράμμα "eco" εμφανιζόταν στο αλφαριθμητικό δύο φορές τότε θα ίσχυε $a_{eco} = 2$. Αντίστοιχα εφόσον εμφανιζόταν τρεις φορές θα ίσχυε $a_{eco} = 3$. Αφού υπολογίσει το διάνυσμα $\vec{A}$, η Αλίκη στη συνέχεια κρυπτογραφεί όλα τα διανύσματα και εφαρμόζει πάνω τους την συνάρτηση κατακερματισμού. Παράλληλα δημιουργεί κρυπτογραφημένες εκδόσεις των κλειδιών αναγνώρισης κάθε εγγραφής, και τελικά στέλνει όλα τα δεδομένα στην Κάρολ.

3. Ομοίως με την Αλίκη, ο Μπομπ εκτελεί επακριβώς για τα δικά του δεδομένα την ίδια διαδικασία που περιγράφηκε στο βήμα 2, και στέλνει τα αποτελέσματα στην Κάρολ.

4. Τέλος η Κάρολ, για κάθε ξεχωριστό ζεύγος κλειδιών αναγνώρισης εγγραφών, υπολογίζει την διαφορά των δύο διανυσμάτων αφαιρώντας το ένα από το άλλο. Συγκεκριμένα υπολογίζει την ένταση της διαφοράς των δύο διανυσμάτων η οποία, για δύο διανύσματα $\vec{A}$ και $\vec{B}$, δίνεται από τον τύπο:

$$\left\|\vec{D}\right\| = \sqrt{\sum_{i=aaa}^{999} (a_i - b_i)^2}$$

Κατόπιν η ένταση της διαφοράς των δύο διανυσμάτων συγκρίνεται με μια προκαθορισμένη τιμή κατωφλίου. Εάν η ένταση είναι μικρότερη της τιμής κατωφλίου τότε οι δύο εγγραφές θεωρείται ότι αντιστοιχούν στην ίδια οντότητα, αλλιώς θεωρείται ότι δεν ταιριάζουν. Σύμφωνα με τον J. Hylton [23], κατόπιν αρκετών πειραμάτων η τιμή κατωφλίου *T* που χρησιμοποιείται για την

σύγκριση δύο αλφαριθμητικών με συνολικό αριθμό Τρι-γραμμάτων *n*, δίνεται από τον τύπο:

$$T = 2,486 + 0,025n$$

Αφού η Κάρολ υπολογίσει όλες τις διαφορές, αποστέλλει τα αποτελέσματα της σύγκρισης στην Αλίκη και τον Μπομπ που θα κρίνουν πώς θα τα χρησιμοποιήσουν.

## 3.4 Σύγκριση Ομοιότητας με Χρήση της Απόστασης Levenshtein

Το τρίτο πρωτόκολλο εφαρμόζει την απόσταση Levenshtein προκειμένου να επιτύχει την σύγκριση της ομοιότητας των γνωρισμάτων. Ο αλγόριθμος που χρησιμοποιεί η απόσταση Levenshtein για την σύγκριση των αλφαριθμητικών παρουσιάζει μια ενδιαφέρουσα ιδιαιτερότητα. Έχει την ικανότητα να εφαρμόζει με απόλυτη επιτυχία την σύγκριση της ομοιότητας μη κρυπτογραφημένων δεδομένων, αλλά αντιμετωπίζει προβλήματα όταν επεξεργάζεται κρυπτογραφημένα δεδομένα. Όπως προαναφέρθηκε στην ενότητα 2.1.2, ο αλγόριθμος υπολογίζει την απόσταση δύο αλφαριθμητικών μετρώντας το πλήθος των πράξεων που πρέπει να γίνουν προκειμένου να μετατραπεί το ένα στο άλλο. Όταν όμως εφαρμόζεται κάποια διαδικασία κρυπτογράφησης σε δύο αλφαριθμητικά που έχουν μια συγκεκριμένη απόσταση μεταξύ τους, κατά πάσα πιθανότητα τα κρυπτογραφημένα αλφαριθμητικά που θα προκύψουν θα έχουν τελείως διαφορετική απόσταση. Για παράδειγμα, παρακάτω παρατίθενται δύο αλφαριθμητικά και η κρυπτογραφημένη και κατακερματισμένη εκδοχή τους:

Johnson ↔ d184ce4cf765d0c9019b34a57d78d274

Johnsen ↔ e5d624ba091ec168f5ed5764aa974378

Όπως φαίνεται από το παράδειγμα, ενώ τα δύο αλφαριθμητικά έχουν απόσταση *Levenshtein = 1*, οι κρυπτογραφημένες μορφές τους έχουν απόσταση *Levenshtein = 25*.

Προκειμένου να ξεπεραστεί το παραπάνω πρόβλημα εφαρμόστηκε η εξής διαδικασία. Κάθε αλφαριθμητικό διασπάται στους χαρακτήρες από τους οποίους

αποτελείται. Κατόπιν εφαρμόζεται σε κάθε χαρακτήρα ξεχωριστά η συνάρτηση κατακερματισμού. Τέλος εφαρμόζεται ο αλγόριθμος της απόστασης Levenshtein, συγκρίνοντας χαρακτήρα με χαρακτήρα. Αυτό είναι δυνατόν να συμβεί γιατί κάθε χαρακτήρας θα έχει πάντα την ίδια σύνοψη αν εφαρμόζεται σε αυτόν η ίδια συνάρτηση κατακερματισμού με το ίδιο κλειδί. Εάν λοιπόν οι κρυπτογραφημένες μορφές δύο χαρακτήρων έχουν μηδενική απόσταση Levenshtein, τότε αντιπροσωπεύουν τον ίδιο χαρακτήρα. Σε οποιαδήποτε άλλη περίπτωση αντιπροσωπεύουν διαφορετικούς χαρακτήρες. Τελικά η πραγματική απόσταση μεταξύ δύο αλφαριθμητικών υπολογίζεται αφού έχουν συγκριθεί μια προς μια οι συνόψεις των χαρακτήρων τους.

Προηγουμένως αναφέρθηκε ότι σε κάθε χαρακτήρα εφαρμόζεται μόνο η συνάρτηση κατακερματισμού και όχι ο κρυπτογραφικός αλγόριθμος. Αυτό συμβαίνει διότι ο πηγαίος κώδικας που υλοποιεί το πρωτόκολλο αντιμετώπιζε πρόβλημα υπερχείλισης μνήμης όταν σε κάθε χαρακτήρα εφαρμοζόταν και η μέθοδος κρυπτογράφησης. Για τον λόγο αυτό προτιμήθηκε να εφαρμοστεί μονό η συνάρτηση κατακερματισμού, με την οποία ο κώδικας μπορούσε να εκτελεστεί ανεμπόδιστα. Μάλιστα αξίζει να αναφερθεί ότι η συγκεκριμένη διαδικασία δεν αντιμετωπίζει προβλήματα ασφάλειας, αφού η συνάρτηση κατακερματισμού είναι μονόδρομη, οπότε κανείς δεν μπορεί να αποκαλύψει τα πραγματικά δεδομένα που αντιπροσωπεύει η σύνοψη. Παρόλα αυτά η μέθοδος κρυπτογράφησης χρησιμοποιείται κατά την διάρκεια του πρωτοκόλλου για την κρυπτογράφηση των κλειδιών αναγνώρισης των εγγραφών.

Στην συνέχεια παρατίθενται τα βήματα του πρωτοκόλλου το οποίο υλοποιεί τον αλγόριθμο της απόστασης Levenshtein και για τους χαρακτήρες, αλλά και για ολόκληρα τα αλφαριθμητικά.

1. Ομοίως με τα προηγούμενα πρωτόκολλα, η Αλίκη και ο Μπομπ αποφασίζουν από κοινού για το μυστικό κλειδί κρυπτογράφησης, καθορίζουν τον αλγόριθμο κρυπτογράφησης που θα εφαρμόσουν, ορίζουν την μονόδρομη μαθηματική συνάρτηση κατακερματισμού, και φέρουν τα δεδομένα τους στην ίδια τυποποιημένη μορφή.
2. Κατόπιν η Αλίκη διασπά όλα τα αλφαριθμητικά που περιέχονται στα πεδία $r_{i,l}$ στους χαρακτήρες τους, σηματοδοτώντας τα σημεία όπου λήγει κάθε πεδίο.

Έπειτα εφαρμόζει την συνάρτηση κατακερματισμού σε όλα τα δεδομένα, και τα αποστέλλει στην Κάρολ μαζί με τις κρυπτογραφημένες εκδοχές των κλειδιών αναγνώρισης των εγγραφών.

3. Ο Μπομπ, όμοια με την Αλίκη, εκτελεί την ίδια διαδικασία για τα δικά του δεδομένα που περιέχονται στα $rec_{j,j}$, και στέλνει τα αποτελέσματα στην Κάρολ.
4. Τέλος η Κάρολ, για κάθε ξεχωριστό ζεύγος κλειδιών αναγνώρισης εγγραφών, υπολογίζει την απόσταση Levenshtein, και αν δεν ξεπερνά ένα προκαθορισμένο κατώφλι $\theta$, το ζεύγος θεωρείται ότι ταιριάζει. Στην συνέχεια αποστέλλει όλα τα αποτελέσματα στην Αλίκη και τον Μπομπ.

Αξίζει να σημειωθεί ότι επιτυχημένο ταίριασμα γνωρισμάτων μπορεί να επιτευχθεί αν οριστεί ένα κατώφλι δύο ή τριών αντιμεταθέσεων. Όμως κατά αυτόν τον τρόπο υπάρχει η πιθανότητα να ταιριάξουν γνωρίσματα με μικρά αλφαριθμητικά, των οποίων οι εγγραφές αντιπροσωπεύουν διαφορετικές οντότητες. Για τον λόγο αυτό ορίστηκε το κάτωθι κατώφλι:

$$\theta = \frac{\mathrm{T}}{|s_1|+|s_2|}, \text{ όπου}$$

- $T$ είναι το πλήθος των αντιμεταθέσεων,
- $|s_1|$, $|s_2|$ είναι το μήκος των δύο αλφαριθμητικών αντίστοιχα.

Πλέον αν καθοριστεί $\theta \leq 1/5$, το πρωτόκολλο μπορεί να ταιριάξει σωστά τα γνωρίσματα που αντιστοιχούν στην ίδια οντότητα και παράλληλα αποφεύγει τα λάθος ταιριάσματα.

## 3.5 Σύγκριση Ομοιότητας με Χρήση της Απόστασης Jaro-Winkler

Το τελευταίο πρωτόκολλο που υλοποιήθηκε κατά την διάρκεια της εργασίας κάνει χρήση του αλγορίθμου της απόστασης Jaro-Winkler, προκειμένου να προβεί στην διασύνδεση των γνωρισμάτων. Ο συγκεκριμένος αλγόριθμος παρουσιάζει την ίδια

ιδιαιτερότητα με τον αλγόριθμο της απόστασης Levenshtein όσον αφορά την σύγκριση κρυπτογραφημένων δεδομένων. Προκειμένου λοιπόν να αντιμετωπίσει ορθά τα κρυπτογραφημένα δεδομένα εφαρμόζει την ίδια διαδικασία που περιγράφθηκε στην ενότητα 3.4, σπάζοντας τα αλφαριθμητικά στους χαρακτήρες τους. Παρακάτω παρατίθενται τα βήματα που ακολουθεί το πρωτόκολλο.

1. Παρόμοια με τα υπόλοιπα πρωτόκολλα, η Αλίκη και ο Μπομπ καθορίζουν το μυστικό κλειδί και τον αλγόριθμο κρυπτογράφησης που θα εφαρμόσουν, ορίζουν την μονόδρομη μαθηματική συνάρτηση κατακερματισμού, και φέρουν τα δεδομένα τους στην ίδια τυποποιημένη μορφή.
2. Στη συνέχεια η Αλίκη, ομοίως με το πρωτόκολλο της ενότητας 3.4, διασπά όλα τα αλφαριθμητικά των πεδίων $r_{i,i}$ στους χαρακτήρες τους, οριοθετώντας την λήξη κάθε πεδίου. Έπειτα συνοψίζει όλα τα δεδομένα, και τα αποστέλλει στην Κάρολ μαζί με τις κρυπτογραφημένες εκδόσεις των κλειδιών αναγνώρισης των εγγραφών.
3. Αντίστοιχα ο Μπομπ, εκτελεί την ίδια διαδικασία για τα δικά του δεδομένα που περιέχονται στα $rec_{j,j}$, και στέλνει τα αποτελέσματα στην Κάρολ.
4. Κατόπιν η Κάρολ, για κάθε ξεχωριστό ζεύγος κλειδιών αναγνώρισης εγγραφών, υπολογίζει την απόσταση Jaro-Winkler όπως περιγράφθηκε στην ενότητα 2.1.3, και αν αυτή ξεπερνά ένα προκαθορισμένο κατώφλι $\theta$, το ζεύγος θεωρείται ότι ταιριάζει. Τέλος η Κάρολ αποστέλλει τα αποτελέσματα στην Αλίκη και τον Μπομπ.

Είναι προφανές ότι το κατώφλι παίρνει τιμές $0 \leq \theta \leq 1$, όπου για $\theta = 1$ υπάρχει το απόλυτο ταίριασμα. Μάλιστα, κατόπιν αρκετών πειραμάτων, έγινε εμφανές ότι ο αλγόριθμος πετυχαίνει ορθά την διασύνδεση εγγραφών όταν το κατώφλι είναι $\theta = 0,8$.

## 3.6 Σύγκριση Ομοιότητας με Εφαρμογή Blocking

Τα πρωτόκολλα που περιγράφθηκαν προηγουμένως εφαρμόζουν τις διαδικασίες της διασύνδεσης γνωρισμάτων σε όλα τα πιθανά ζεύγη εγγραφών των δύο συνόλων

δεδομένων. Αυτό σημαίνει ότι κατά την επεξεργασία πολύ μεγάλων συνόλων δεδομένων, η διαδικασία της σύγκρισης εφαρμόζεται σε μεγάλο πλήθος εγγραφών, με αποτέλεσμα να καταναλώνονται υπερβολικά πολλοί υπολογιστικοί πόροι. Ένας τρόπος για να ελαχιστοποιηθεί το υπολογιστικό κόστος είναι να μειωθεί το πλήθος των συγκρίσεων μεταξύ εγγραφών που κατά πάσα βεβαιότητα αντιστοιχούν σε διαφορετικές οντότητες. Σε αυτήν την κατεύθυνση η πιο αποδοτική λύση είναι να εφαρμοστούν τεχνικές Blocking όπως περιγράφθηκαν στην ενότητα 2.2.

Ουσιαστικά κατά την εφαρμογή του Blocking χρησιμοποιούνται τα ίδια πρωτόκολλα σύγκρισης γνωρισμάτων που περιγράφθηκαν προηγουμένως, αλλά η σύγκριση γίνεται μόνο μεταξύ εγγραφών που ανήκουν στο ίδιο μπλοκ. Μάλιστα όπως προαναφέρθηκε στην ενότητα 2.2.1, προκειμένου η σύγκριση να δώσει πιο σωστά αποτελέσματα, είναι δυνατόν να γίνουν επαναληπτικές συγκρίσεις με διαφορετικό κάθε φορά κλειδί Blocking.

Τα βήματα λοιπόν που ακολουθεί οποιοδήποτε από τα πρωτόκολλα που παρουσιάστηκαν στις ενότητες 3.2 ως 3.5, όταν εφαρμόζει πρώτα μια τεχνική Blocking, διαφαίνονται στην συνέχεια.

1. Αρχικά η Αλίκη και ο Μπομπ συμφωνούν από κοινού για το μυστικό κλειδί κρυπτογράφησης και τον αλγόριθμο κρυπτογράφησης που θα εφαρμόσουν, ορίζουν την μονόδρομη μαθηματική συνάρτηση κατακερματισμού, και φέρουν τα δεδομένα τους στην ίδια τυποποιημένη μορφή.
2. Στην συνέχεια καθορίζουν την τεχνική Blocking που πρόκειται να ακολουθήσουν και παράλληλα συνεννοούνται για το ποιο θα είναι το κλειδί Blocking. Επίσης αποφασίζουν αν θα γίνουν επαναληπτικές συγκρίσεις, και ποιο θα είναι το κλειδί Blocking κάθε φορά.
3. Έπειτα η Αλίκη προετοιμάζει τα δεδομένα της ανάλογα με το πρωτόκολλο που εφαρμόζεται, και οργανώνει τις εγγραφές της σε μπλοκ. Κατόπιν, ανάλογα με το πρωτόκολλο, εφαρμόζει στα δεδομένα την συνάρτηση κρυπτογράφησης και την συνάρτηση κατακερματισμού ή μόνο την συνάρτηση κατακερματισμού, και δημιουργεί κρυπτογραφημένες εκδόσεις των κλειδιών αναγνώρισης των εγγραφών. Παράλληλα δημιουργεί κρυπτογραφημένες εκδόσεις όλων των

κλειδιών αναγνώρισης των μπλοκ, και αποστέλλει όλα τα στοιχεία στην Κάρολ.

4. Αντίστοιχα ο Μπομπ εκτελεί την ίδια διαδικασία για τα δικά του δεδομένα και στέλνει τα αποτελέσματα με την σειρά του στην Κάρολ.
5. Τέλος η Κάρολ, για κάθε ξεχωριστό ζεύγος κλειδιών αναγνώρισης εγγραφών οι οποίες έχουν το ίδιο κλειδί αναγνώρισης μπλοκ, εφαρμόζει τον αλγόριθμο σύγκρισης της ομοιότητας των εγγραφών ανάλογα με το πρωτόκολλο, και αποστέλλει τα αποτελέσματα στην Αλίκη και τον Μπομπ. Αξίζει να σημειωθεί ότι κάποιες εγγραφές ενδέχεται να μην ανήκουν σε κανένα μπλοκ λόγω του ότι το πεδίο που ορίστηκε ως κλειδί μπορεί να ήταν κενό. Στην περίπτωση αυτή οι συγκεκριμένες εγγραφές θεωρούνται ότι ανήκουν στο ίδιο μπλοκ και συγκρίνονται μεταξύ τους.

# Κεφάλαιο 4

# Επεξήγηση Προγραμμάτων

Σε αυτό το κεφάλαιο γίνεται η επεξήγηση των προγραμμάτων που υλοποιήθηκαν κατά την διάρκεια της έρευνας, προκειμένου να επιτευχθεί και να εξεταστεί η εφαρμογή των πρωτοκόλλων που περιγράφθηκαν στο κεφάλαιο 3. Όλα τα προγράμματα είναι γραμμένα στην γλώσσα προγραμματισμού Perl [56]. Όπως έχει αναφερθεί και σε προηγούμενη εργασία [51], ο λόγος που επιλέχθηκε αυτή η γλώσσα προγραμματισμού αντί κάποιας άλλης όπως οι ευρέως χρησιμοποιούμενες C/C++ και Java, είναι ότι η σύνταξη της είναι απλούστερη και, όσον αφορά τα προγράμματα που επρόκειτο να υλοποιηθούν, είναι πιο πρακτική και λειτουργική. Αυτό ισχύει επειδή η Perl επιτυγχάνει τη σύγκριση δύο ή περισσότερων αλφαριθμητικών με χρήση απλών εντολών, σε αντίθεση με άλλες γλώσσες προγραμματισμού οι οποίες απαιτούν την υλοποίηση πολύπλοκου κώδικα.

Συνολικά για τις ανάγκες της εργασίας υλοποιήθηκαν 20 προγράμματα όπως προαναφέρθηκε στην ενότητα 1.3. Η λογική και το γενικό πλαίσιο ορισμένων εξ αυτών είναι παρόμοια και αλλάζουν μόνο συγκεκριμένα σημεία, που αφορούν το κάθε πρωτόκολλο ξεχωριστά. Για τον λόγο αυτό η περιγραφή ορισμένων προγραμμάτων θα είναι κοινή, τονίζοντας παράλληλα τις διαφορές τους. Επίσης, εκτός από τα προγράμματα που παράχθηκαν κατά την διάρκεια της έρευνας, χρησιμοποιήθηκε και μια βιβλιοθήκη προγραμμάτων [57] η οποία έχει υλοποιηθεί για την γλώσσα προγραμματισμού Perl και μπορεί να την χρησιμοποιήσει ελεύθερα κάθε χρήστης. Στην συνέχεια του κεφαλαίου παρουσιάζονται τα προγράμματα προετοιμασίας των δεδομένων, κρυπτογράφησης, σύγκρισης και οργάνωσης σε μπλοκ.

## 4.1 Προγράμματα Προετοιμασίας των Δεδομένων

Τα προγράμματα προετοιμασίας των δεδομένων είναι στην ουσία τα προγράμματα που χρησιμοποιούν η Αλίκη και ο Μπομπ, σύμφωνα με το πλαίσιο που περιγράφθηκε στην ενότητα 3.1. Είναι δηλαδή τα προγράμματα που επεξεργάζονται τα δεδομένα με κατάλληλο τρόπο, ώστε να επιτευχθεί η σύγκριση τους ανάλογα με το πρωτόκολλο που εφαρμόζεται. Όσον αφορά το Blocking, επειδή οι αλλαγές που έγιναν στα προγράμματα για να εφαρμοστεί είναι πανομοιότυπες, θα γίνει μια γενική αναφορά στις μεταβολές που επέφερε το Blocking στον κώδικα. Επίσης αξίζει να σημειωθεί ότι χρησιμοποιήθηκε το ίδιο πρόγραμμα προετοιμασίας για τα πρωτόκολλα που κάνουν χρήση των δύο μετρικών αποστάσεως. Αυτό συμβαίνει διότι, όπως περιγράφθηκε στις ενότητες 3.4 και 3.5, τα δύο πρωτόκολλα επεξεργάζονται με τον ίδιο τρόπο τα δεδομένα. Συνολικά υλοποιήθηκαν τέσσερα προγράμματα προετοιμασίας, καθώς και οι αντίστοιχες εκδόσεις τους για εφαρμογή του Blocking:

1. **Πρόγραμμα create_ngrams/create_ngrams_block**: Υπολογίζει τα Ν-γράμματα και το δυναμοσύνολο τους για κάθε πεδίο των εγγραφών ενός συνόλου δεδομένων.
2. **Πρόγραμμα create_ngrams_mod/create_ngrams_mod_block:** Υπολογίζει τα Ν-γράμματα και μόνο τις δύο υπο-λίστες για κάθε πεδίο των εγγραφών ενός συνόλου δεδομένων.
3. **Πρόγραμμα create_trigrams/create_trigrams_block**: Υπολογίζει τα Τρι-γράμματα καθώς και τον αριθμό των εμφανίσεων τους για κάθε πεδίο των εγγραφών ενός συνόλου δεδομένων.
4. **Προγράμματα create_edit_dist / create_edit_dist_block - create_jaro / create_jaro_block:** προετοιμάζουν τα δεδομένα για να εφαρμοστούν σε αυτά οι αλγόριθμοι σύγκρισης με μετρικά απόστασης.

### 4.1.1 Πρόγραμμα Υπολογισμού Ν-Γραμμάτων

Το πρόγραμμα "create_ngrams.plx" υλοποιήθηκε στα πλαίσια προηγούμενης εργασίας [51]. Το συγκεκριμένο πρόγραμμα διαβάζει ένα αρχείο και παράγει τελικά ένα αρχείο με τα Ν-γράμματα όλων των πεδίων του αρχείου που διάβασε, εκτός του πεδίου που περιέχει τον κωδικό της εκάστοτε εγγραφής. Ενδεικτικά δύο αρχεία εισόδου που χρησιμοποιήθηκαν είναι τα "input1.csv" και "input2.csv". Αυτά δημιουργήθηκαν χρησιμοποιώντας την εφαρμογή "generate.py" του λογισμικού Febrl [9], και οι περισσότερες εγγραφές τους είναι παρεμφερείς ώστε στη συνέχεια να γίνει η σύγκριση τόσο των κοινών στοιχείων, όσο και των μη κοινών. Αντιστοίχως, μετά το πέρας της εκτέλεσης του προγράμματος εξάγονται τα αρχεία "outinput1.csv" και "outinput2.csv".

Το αρχείο εισόδου αποτελείται από έναν κωδικό της εγγραφής, δηλαδή το κλειδί αναγνώρισης της, και διάφορα πεδία τα οποία χωρίζονται μεταξύ τους με κόμμα ελληνικό (,). Ενδεικτικά ορισμένα από αυτά περιλαμβάνουν ονόματα, διευθύνσεις, τηλέφωνα. Για όλα τα πεδία κάθε εγγραφής, εκτός από αυτό που περιλαμβάνει τον κωδικό, δημιουργείται ένα ενδιάμεσο αρχείο που ονομάζεται "output.csv". Το αρχείο αυτό περιέχει εγγραφές με στήλες τον κωδικό, ένα ερωτηματικό (;), τον αύξοντα αριθμό (α/α) του πεδίου, ένα κενό και το πεδίο. Δηλαδή για κάθε πεδίο δημιουργείται μία εγγραφή. Επομένως για $r$ εγγραφές και $n$ πεδία δημιουργούνται $r \times n$ εγγραφές. Από κάθε τέτοια εγγραφή κατόπιν παράγονται τα Ν-γράμματα του πεδίου στο αρχείο outinput. Στον παρακάτω πίνακα διακρίνεται ένα τμήμα του αρχείου 'output.csv' το οποίο αναφέρεται στα πεδία μιας εγγραφής.

| **Κωδικός** | **;** | **α/α** | **Τιμή Πεδίου** |
|---|---|---|---|
| $r_5$ | ; | 1 | nicholas |
| $r_5$ | ; | 2 | mercorella |
| $r_5$ | ; | 3 | 84 |
| $r_5$ | ; | 4 | fitchettstreet |
| $r_5$ | ; | 5 | taronga |
| $r_5$ | ; | 6 | toorak |
| $r_5$ | ; | 7 | 2560 |
| $r_5$ | ; | 8 | qld |
| $r_5$ | ; | 9 | 19061026 |
| $r_5$ | ; | 10 | 22 |
| $r_5$ | ; | 11 | 0243222792 |
| $r_5$ | ; | 12 | 8111523 |
| $r_5$ | ; | 13 | 1 |

Πίνακας 4-1: Παράδειγμα μορφής αρχείου output.csv

Το πρόγραμμα παίρνει σαν παραμέτρους το όνομα του αρχείου εισόδου, το όνομα του αρχείου εξόδου και το μήκος των Ν-γραμμάτων, δηλαδή τον αριθμό Ν. Στην αρχή ανοίγει το αρχείο εισόδου και διαβάζει μία μία τις εγγραφές του. Τότε τις καθαρίζει από τα κενά, μετατρέπει όλους τους χαρακτήρες των πεδίων σε πεζά και χρησιμοποιώντας ως χαρακτήρα οριοθέτησης το κόμμα (,) σπάει τα πεδία του σε ένα πίνακα. Από αυτόν τον πίνακα γράφονται στο ενδιάμεσο αρχείο "output" οι εγγραφές με την μορφή: "κωδικός ; α/α πεδίο".

Κατόπιν κλείνουν τα αρχεία και πλέον ανοίγουν σαν είσοδος το ενδιάμεσο αρχείο που έχει δημιουργηθεί και σαν έξοδος το "outinput". Δημιουργούνται τα N-γράμματα του πεδίου σε πίνακα, γίνεται η ταξινόμηση τους και διαγράφονται τα πολλαπλά έτσι ώστε κάθε ένα να εμφανίζεται μια ακριβώς φορά όπως προαναφέρθηκε στο δεύτερο βήμα του πρωτοκόλλου της ενότητας 3.2. Έπειτα δημιουργείται ένας πίνακας κατακερματισμού που περιέχει N-γράμματα για κλειδιά και αριθμούς 1,2,3,... αντίστοιχα για τιμές. Καλείται η υπό-ρουτίνα "powerset_iterate" η οποία έχει ως παράμετρο μια μεταβλητή αναφοράς στον πίνακα κατακερματισμού. Αυτή κάνοντας χρήση αναδρομής δημιουργεί μια δομή υποσυνόλων των αντίστοιχων N-γραμμάτων, δηλαδή το δυναμό-σύνολο. Μετά καλείται η υπό-ρουτίνα "sos_as_string" η οποία πρόκειται να δημιουργήσει σαν αλφαριθμητικά τις υπο-λίστες των N-γραμμάτων, εκτός του κενού συνόλου.

Όλες οι παραπάνω υπο-λίστες γράφονται στο αρχείο "outinput" με τη μορφή "κωδικός ; α/α N-γράμμα". Τα πεδία που είναι κενά και αυτά που ο αριθμός των χαρακτήρων τους είναι μικρότερος ή ίσος με τον αριθμό N που έχουμε δηλώσει στην αρχή, γράφονται όπως είναι στην εγγραφή. Δηλαδή αυτό που γίνεται είναι να μεταφέρεται αυτούσιο το πεδίο. Στον παρακάτω πίνακα παρουσιάζεται ένα παράδειγμα. Θεωρείται ότι το πεδίο 2 του κωδικού $r_{256}$ είναι "George". Τότε παράγονται τα ακόλουθα N-γράμματα : ('ge', 'eo', 'or', 'rg', 'ge'), και μετά την ταξινόμηση και την δημιουργία των υπο-λιστών έχουμε τις παρακάτω τελικές εγγραφές στο αρχείο outinput:

| Κωδικός | ; | α/α | N-γράμμα |
|---|---|---|---|
| $r_{256}$ | ; | 2 | eo |
| $r_{256}$ | ; | 2 | eoge |
| $r_{256}$ | ; | 2 | eogeor |
| $r_{256}$ | ; | 2 | eogeorrg |
| $r_{256}$ | ; | 2 | eogerg |
| $r_{256}$ | ; | 2 | eoor |
| $r_{256}$ | ; | 2 | eoorrg |
| $r_{256}$ | ; | 2 | eorg |
| $r_{256}$ | ; | 2 | ge |
| $r_{256}$ | ; | 2 | geor |
| $r_{256}$ | ; | 2 | georrg |
| $r_{256}$ | ; | 2 | gerg |
| $r_{256}$ | ; | 2 | or |
| $r_{256}$ | ; | 2 | orrg |
| $r_{256}$ | ; | 2 | rg |

Πίνακας 4-2: Παράδειγμα μορφής αρχείου outinput για N-γράμματα

Τέλος προκειμένου να γίνει η σύγκριση των αρχείων με τα N-γράμματα των δύο αρχείων που χρησιμοποιήθηκαν σαν είσοδοι, δημιουργούνται και δύο αρχεία κλειδιών που περιέχουν μόνο τους κωδικούς σε ταξινομημένη σειρά. Παράλληλα τα αρχεία εξόδου output που έχουν τα N-γράμματα είναι ταξινομημένα κατά κωδικό, α/α πεδίου και το ίδιο το N-γράμμα. Οι ταξινομήσεις αυτές γίνονται με βάση τα αλφαριθμητικά.

Στον παρακάτω πίνακα φαίνεται τμήμα του αρχείου με τα ταξινομημένα κλειδιά του συνόλου δεδομένων input1.csv.

| |
|---|
| rec-0-org |
| rec-1-org |
| rec-10-org |
| rec-11-org |
| rec-12-org |
| rec-13-org |
| rec-14-org |
| rec-15-org |
| rec-16-org |
| rec-17-org |

Πίνακας 4-3: Παράδειγμα μορφής αρχείου skeyinput1.csv

### 4.1.2 Τροποποιήσεις Κώδικα Υπολογισμού Ν-Γραμμάτων

Το πρόγραμμα "create_ngrams_mod" χρησιμοποιεί τον ίδιο πηγαίο κώδικα με το αρχικό πρωτόκολλο, εκτός από ορισμένα σημεία που έχουν αλλαχθεί για να πληρούν τις νέες προϋποθέσεις. Το συγκεκριμένο πρόγραμμα, εφόσον δεν παράγει δυναμοσύνολα Ν-γραμμάτων, δεν χρειάζεται να δημιουργήσει τον πίνακα κατακερματισμού και να καλέσει μέσα σε αυτόν την υπο-ρουτίνα "powerset_iterate". Αντί αυτών, εφαρμόζει δύο επαναληπτικούς βρόχους "for" τον έναν μέσα στον άλλον όπου δημιουργείται η υπο-λίστα Ν-γραμμάτων με μήκος δύο. Η υπο-λίστα με μήκος ένα, δηλαδή αυτή που περιέχει κάθε Ν-γράμμα ξεχωριστά, έχει ήδη συσταθεί όπως περιγράφθηκε στην ενότητα 4.1.1. Είναι φανερό ότι τα προηγούμενα εφαρμόζονται μόνο στα δύο πεδία των εγγραφών που έχει προκαθορίσει ο χρήστης. Εφαρμόζοντας την παραπάνω διαδικασία το πρόγραμμα κερδίζει και σε χώρο, αφού υπάρχουν λιγότεροι συνδυασμοί Ν-γραμμάτων, αλλά και σε χρόνο αφού, εκτός των άλλων, έχει αφαιρεθεί η υπο-ρουτίνα "powerset_iterate".

Τέλος, σύμφωνα με τον κώδικα που περιγράφεται στην ενότητα 4.1.1, οι παραπάνω υπο-λίστες γράφονται στο αρχείο "outinput" με τη μορφή "κωδικός ; α/α N-γράμμα". Τα πεδία που είναι κενά και αυτά που ο αριθμός των χαρακτήρων τους είναι μικρότερος ή ίσος με τον αριθμό N που έχουμε δηλώσει στην αρχή, γράφονται όπως είναι στην εγγραφή.

Αντίστοιχα με το παράδειγμα "George" του κωδικού $r_{256}$ που δόθηκε στην ενότητα 4.1.1, οι τελικές εγγραφές στο πεδίο "ouinput" φαίνονται στον πίνακα 4-4.

| **Κωδικός** | **;** | **α/α** | **N-γράμμα** |
|---|---|---|---|
| $r_{256}$ | ; | 2 | eo |
| $r_{256}$ | ; | 2 | eoge |
| $r_{256}$ | ; | 2 | eoor |
| $r_{256}$ | ; | 2 | eorg |
| $r_{256}$ | ; | 2 | ge |
| $r_{256}$ | ; | 2 | geor |
| $r_{256}$ | ; | 2 | gerg |
| $r_{256}$ | ; | 2 | or |
| $r_{256}$ | ; | 2 | orrg |
| $r_{256}$ | ; | 2 | rg |

Πίνακας 4-4: Παράδειγμα μορφής τροποποιημένου αρχείου outinput

### 4.1.3 Πρόγραμμα Υπολογισμού Τρι-Γραμμάτων

Το πρωτόκολλο χρήσης Τρι-γραμμάτων προετοιμάζει τα δεδομένα με το πρόγραμμα "create_trigrams.plx". Το συγκεκριμένο πρόγραμμα διαβάζει ένα αρχείο και παράγει τελικά ένα άλλο αρχείο με τα Τρι-γράμματα όλων των πεδίων του αρχείου που διάβασε

και τον αριθμό των εμφανίσεων τους, εκτός του πεδίου που περιέχει τον κωδικό της εκάστοτε εγγραφής. Τα αρχεία εισόδου που χρησιμοποιήθηκαν, τα ενδιάμεσα αρχεία, καθώς και τα αρχεία που εξάγονται είναι παρόμοια με αυτά της ενότητας 4.1.1.

Το πρόγραμμα παίρνει σαν παραμέτρους το όνομα του αρχείου εισόδου, το όνομα του αρχείου εξόδου και το μήκος των Τρι-γραμμάτων, δηλαδή τον αριθμό 3. Στην αρχή, ομοίως με το πρόγραμμα της ενότητας 4.1.1, ανοίγει το αρχείο εισόδου και διαβάζει μία μία τις εγγραφές του. Τότε τις καθαρίζει από τα κενά, μετατρέπει όλους τους χαρακτήρες των πεδίων σε πεζά και χρησιμοποιώντας ως χαρακτήρα οριοθέτησης το κόμμα (,) σπάει τα πεδία του σε ένα πίνακα. Από αυτόν τον πίνακα γράφονται στο ενδιάμεσο αρχείο 'output' οι εγγραφές με την μορφή: "κωδικός ; α/α πεδίο".

Κατόπιν κλείνουν τα αρχεία και πλέον ανοίγουν σαν είσοδος το ενδιάμεσο αρχείο που έχει δημιουργηθεί και σαν έξοδος το "outinput". Δημιουργούνται τα Τρι-γράμματα του πεδίου σε πίνακα, γίνεται η ταξινόμηση τους και μετράται ο αριθμός εμφάνισης τους, έτσι ώστε να είναι γνωστό πόσες φορές επαναλαμβάνεται κάθε Τρι-γράμμα. Δηλαδή κάθε εγγραφή του αρχείου "outinput" έχει τη μορφή "κωδικός/ ;/ α-α πεδίου/ Τρι-γράμμα/ αριθμός εμφάνισης Τρι-γράμματος". Τα πεδία που είναι κενά και αυτά που ο αριθμός των χαρακτήρων τους είναι μικρότερος ή ίσος με τον αριθμό 3, γράφονται όπως είναι στην εγγραφή με την προσθήκη στο τέλος του αριθμού 1, δηλαδή σαν να εμφανίζονται μόνο μια φορά.

Προκειμένου να γίνουν τα προηγούμενα κατανοητά παρουσιάζεται το εξής παράδειγμα. Θεωρείται ότι το πεδίο 2 του κωδικού $r_{256}$ είναι "801150115011". Τότε παράγονται τα Τρι-γράμματα ('801', '011', '115', '150', '501'). Μετά την ταξινόμηση και μέτρηση τους, στο αρχείο "outinput" καταγράφονται οι τελικές εγγραφές που φαίνονται στον πίνακα 4-5.

| Κωδικός | ; | α/α | Τρι-γράμμα | πλήθος εμφανίσεων |
|---|---|---|---|---|
| $r_{256}$ | ; | 2 | 011 | 3 |
| $r_{256}$ | ; | 2 | 115 | 2 |
| $r_{256}$ | ; | 2 | 150 | 2 |
| $r_{256}$ | ; | 2 | 501 | 2 |
| $r_{256}$ | ; | 2 | 801 | 1 |

Πίνακας 4-5: Παράδειγμα μορφής αρχείου outinput για Τρι-γράμματα

Τέλος προκειμένου να γίνει η σύγκριση των Τρι-γραμμάτων των δύο αρχείων που χρησιμοποιήθηκαν σαν είσοδοι, δημιουργούνται και δύο αρχεία κλειδιών που περιέχουν μόνο τους κωδικούς σε ταξινομημένη σειρά. Παράλληλα τα αρχεία εξόδου "output" που έχουν τα Τρι-γράμματα είναι ταξινομημένα κατά κωδικό, α/α πεδίου και το ίδιο το Τρι-γράμμα. Οι ταξινομήσεις αυτές γίνονται με βάση τα αλφαριθμητικά.

### 4.1.4 Πρόγραμμα Προετοιμασίας Μετρικών Απόστασης

Τα προγράμματα "create_edit_dist.plx" και "create_jaro.plx" ουσιαστικά αποτελούν το ίδιο πρόγραμμα το οποίο είναι ιδιαίτερα απλό. Το μόνο που κάνει είναι να διαβάζει ένα .csv αρχείο εισόδου, για παράδειγμα το "input1.csv", όπου τα πεδία των εγγραφών είναι χωρισμένα με ελληνικό κόμμα (,), και να παράγει ένα παρόμοιο .csv αρχείο. Στο παραγόμενο αρχείο τα πεδία είναι πλέον χωρισμένα με τελεία κόμμα (;), έχουν σβηστεί όλα τα κενά και τα κεφαλαία γράμματα έχουν γίνει πεζά.

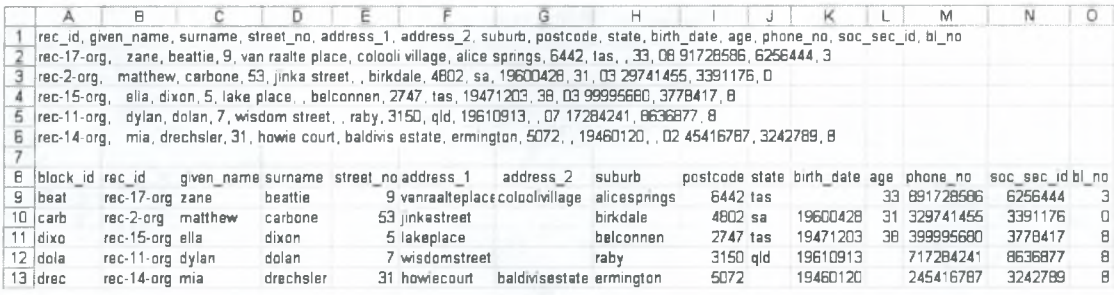
Αξίζει να σημειωθεί ότι στην περίπτωση που δεν πρόκειται να εφαρμοστεί κρυπτογράφηση, τα αρχεία είναι έτοιμα για να τα επεξεργαστούν τα προγράμματα σύγκρισης της ομοιότητας με χρήση των αλγορίθμων Levenshtein και Jaro – Winkler. Στην περίπτωση που θα εφαρμοστεί κρυπτογράφηση, τότε τα αλφαριθμητικά θα πρέπει

να διασπαστούν στους χαρακτήρες τους, μια διαδικασία που θα γίνει στο πρόγραμμα κρυπτογράφησης.

### 4.1.5 Προγράμματα Εφαρμογής Blocking

Σε όλα τα προγράμματα προετοιμασίας των δεδομένων έγιναν κοινές αλλαγές προκειμένου να είναι σε θέση να δημιουργήσουν μπλοκ εγγραφών. Στην εφαρμογή επιλέχθηκε να υλοποιηθεί το απλό Blocking όπως περιγράφθηκε στην ενότητα 2.2.1. Πλέον τα προγράμματα διαβάζουν το αρχείο .csv, για παράδειγμα το "input1.csv", όπου τα πεδία είναι χωρισμένα με κόμμα (,), και παράγουν ένα παρόμοιο αρχείο, το "bloutinput1.csv" το οποίο περιέχει ένα επιπλέον πεδίο. Το επιπλέον πεδίο είναι το μπλοκ που έχει καθοριστεί, το οποίο τοποθετείται σαν πρώτο πεδίο και κατόπιν ακολουθούν τα υπόλοιπα πεδία τα οποία πλέον διαχωρίζονται με τελεία κόμμα (;). Εκτός των άλλων, από κάθε πεδίο διαγράφονται όλοι οι κενοί χαρακτήρες.

Προκειμένου να καθοριστεί το πεδίο μπλοκ, επιλέγεται ένα από τα πεδία δηλώνοντας την σειρά που έχει στις εγγραφές, καθώς και ο αριθμός των πρώτων *n* χαρακτήρων του που πρόκειται να αποτελέσουν το κλειδί Blocking. Ο καθορισμός αυτός γίνεται παραμετρικά όταν δίνεται η εντολή εκτέλεσης του προγράμματος. Κατόπιν η σύγκριση θα γίνει μόνο μεταξύ εγγραφών που περιέχουν το ίδιο αλφαριθμητικό στο πρώτο πεδίο του νέου .csv αρχείου. Στο σχήμα 4-1 διακρίνεται ένα στιγμιότυπο του αρχείου "input1.csv" και του παραγόμενου "bloutinput1.csv" όταν έχουν επιλεγεί σαν κλειδί Blocking οι τέσσερις πρώτοι χαρακτήρες του πεδίου επώνυμο. Στην εικόνα φαίνονται οι διαφορές των αρχείων όσον αφορά τον διαχωρισμό των πεδίων.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | rec_id, given_name, surname, street_no, address_1, address_2, suburb, postcode, state, birth_date, age, phone_no, soc_sec_id, bl_no | | | | | | | | | | | | | | |
| 2 | rec-17-org, zane, beattie, 9, van raalte place, colooli village, alice springs, 6442, tas, , 33, 08 91728586, 6256444, 3 | | | | | | | | | | | | | | |
| 3 | rec-2-org, matthew, carbone, 53, jinka street, , birkdale, 4802, sa, 19600428, 31, 03 29741455, 3391176, 0 | | | | | | | | | | | | | | |
| 4 | rec-15-org, ella, dixon, 5, lake place, , belconnen, 2747, tas, 19471203, 38, 03 99995680, 3778417, 8 | | | | | | | | | | | | | | |
| 5 | rec-11-org, dylan, dolan, 7, wisdom street, , raby, 3150, qld, 19610913, , 07 17284241, 8636877, 8 | | | | | | | | | | | | | | |
| 6 | rec-14-org, mia, drechsler, 31, howie court, baldivis estate, ermington, 5072, , 19460120, , 02 45416787, 3242789, 8 | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | |
| 8 | block_id | rec_id | given_name | surname | street_no | address_1 | address_2 | suburb | postcode | state | birth_date | age | phone_no | soc_sec_id | bl_no |
| 9 | beat | rec-17-org | zane | beattie | 9 | vanraaltepla | coloolivillage | alicesprngs | 6442 | tas | | 33 | 891728586 | 6256444 | 3 |
| 10 | carb | rec-2-org | matthew | carbone | 53 | jinkastreet | | birkdale | 4802 | sa | 19600428 | 31 | 329741455 | 3391176 | 0 |
| 11 | dixo | rec-15-org | ella | dixon | 5 | lakeplace | | belconnen | 2747 | tas | 19471203 | 38 | 399995680 | 3778417 | 8 |
| 12 | dola | rec-11-org | dylan | dolan | 7 | wisdomstreet | | raby | 3150 | qld | 19610913 | | 717284241 | 8636877 | 8 |
| 13 | drec | rec-14-org | mia | drechsler | 31 | howiecourt | baldivisestate | ermington | 5072 | | 19460120 | | 245416787 | 3242789 | 8 |

Σχήμα 4-1: Στιγμιότυπο αρχείων “input1.csv” και “bloutinput1.csv”

## 4.2 Προγράμματα Κρυπτογράφησης

Μετά την προ-επεξεργασία των δεδομένων λαμβάνει χώρα η κρυπτογράφηση τους ώστε να διασφαλιστεί η εμπιστευτικότητα τους. Όπως έχει αναφερθεί στην ενότητα 2.3 χρησιμοποιήθηκαν ο κρυπτογραφικός αλγόριθμος Twofish και η συνάρτηση κατακερματισμού MD5. Για τον λόγο αυτό χρησιμοποιήθηκαν τα modules Crypt::Twofish και Digest::MD5 [57], τα οποία αποτελούν Perl επεκτάσεις. Με τον όρο Perl επέκταση, εννοείται ότι τα συγκεκριμένα modules αποτελούν ένα είδος βιβλιοθήκης προγραμμάτων τα οποία αποθηκεύονται στην Perl και ο χρήστης έχει την δυνατότητα να τα χρησιμοποιήσει μέσα από δικά του προγράμματα. Από το module Crypt::Twofish χρησιμοποιήθηκε μόνο η συνάρτηση κρυπτογράφησης. Αντίστοιχα, από το module Digest::MD5 χρησιμοποιήθηκε η συνάρτηση “md5_hex” η οποία επιστρέφει μια σύνοψη του μηνύματος που δέχεται σαν είσοδο σε δεκαεξαδική φόρμα. Το μήκος του επιστρεφόμενου μηνύματος είναι 32 και περιέχει χαρακτήρες από το σύνολο ‘0’ έως ’9’ και ‘a’ έως ‘f’.

Τα προγράμματα κρυπτογράφησης διαχωρίζονται σε αυτά που χρησιμοποιούνται στις εφαρμογές που υπολογίζουν N-γράμματα και Τρι-γράμματα, και σε αυτά που χρησιμοποιούνται στις εφαρμογές χρήσης μετρικών αποστάσεως. Τα πρώτα χρησιμοποιούν και τα δύο modules, ενώ τα δεύτερα χρησιμοποιούν το Crypt::Twofish για τα αναγνωριστικά κάθε εγγραφής, και στα υπόλοιπα δεδομένα εφαρμόζουν μόνο το Digest::MD5. Συνολικά υλοποιήθηκαν έξι αρχεία:

1. **Πρόγραμμα encr_ngrams**: Κρυπτογραφεί τα Ν-γράμματα που έχουν παραχθεί από το πρόγραμμα "create_ngrams.plx".
2. **Πρόγραμμα encr_ngrams_mod:** Κρυπτογραφεί τα Ν-γράμματα που έχουν παραχθεί από το πρόγραμμα " create _ngrams_mod.plx".
3. **Πρόγραμμα encr_trigrams**: Κρυπτογραφεί τα Τρι-γράμματα που έχουν παραχθεί από το πρόγραμμα " create _trigrams.plx".
4. **Πρόγραμμα encr_edit_dist:** Κρυπτογραφεί τα δεδομένα που έχουν παραχθεί από το πρόγραμμα " create _edit_dist.plx".
5. **Πρόγραμμα encr_jaro:** Κρυπτογραφεί τα δεδομένα που έχουν παραχθεί από το πρόγραμμα " create _jaro.plx".
6. **Πρόγραμμα sort_encrypted:** Κάνει την ταξινόμηση των κρυπτογραφημένων δεδομένων.

### 4.2.1 Πρώτη Κατηγορία Προγραμμάτων

Στην πρώτη κατηγορία ανήκουν τα προγράμματα που χρησιμοποιούνται στις εφαρμογές με Ν-γράμματα και Τρι-γράμματα, και ουσιαστικά αποτελούν το ίδιο πρόγραμμα. Στην περίπτωση που εφαρμόζεται Blocking, το πεδίο του μπλοκ, κρυπτογραφείται ως έχει και ακολουθεί ο κωδικός της εγγραφής. Στη συνέχεια το πρόγραμμα διαβάζει τα δεδομένα, που είναι τα Ν-γράμματα και τα Τρι-γράμματα, και το διαχωρίζει με ερωτηματικά (;) και κενά ώστε να πάρει ξεχωριστά κάθε ένα. Κάθε Ν-γράμμα ή Τρι-γράμμα αποτελεί το κείμενο εισόδου προς κρυπτογράφηση που δίνεται σε μια υπο-ρουτίνα "cryptr". Αυτή παίρνει σαν παραμέτρους το κλειδί, το μήκος του κλειδιού και το Ν-γράμμα ή Τρι-γράμμα. Στο σημείο αυτό καλείται η συνάρτηση κρυπτογράφησης του module της TwoFish, που ονομάζεται Encipher($key,$keylength,$plaintext) και κατόπιν η συνάρτηση md5_hex($ciphertext) του module της MD5. Μετά επιστρέφει την σύνοψη, που αναφέρεται ως $digest, η οποία γράφεται σε ένα αρχείο κρυπτογραφημένων δεδομένων με τον κωδικό και το πεδίο σε κάθε εγγραφή. Παραδείγματος χάριν η μορφή του θα είναι:

‘ $r_{224}$ ;1 284a37d79dbdd91e48e0df99ab8ff50a’

Το πρόγραμμα λόγω της κρυπτογράφησης είναι ιδιαίτερα απαιτητικό σε μνήμη, αφού η Perl χρησιμοποιεί διερμηνευτή και όχι μεταγλωττιστή. Για τον λόγο αυτό η ταξινόμηση των δεδομένων γίνεται στο πρόγραμμα “sort_encrypted.plx”.

### 4.2.2 Δεύτερη Κατηγορία Προγραμμάτων

Στην δεύτερη κατηγορία ανήκουν τα προγράμματα των εφαρμογών που κάνουν χρήση των αποστάσεων Levenshtein και Jaro – Winkler. Ουσιαστικά αυτά αποτελούν το ίδιο πρόγραμμα, και η διαφορά τους από τα προηγούμενα είναι ότι στα δεδομένα δεν εφαρμόζεται η συνάρτηση κρυπτογράφησης, και η συνάρτηση κατακερματισμού εφαρμόζεται σε κάθε χαρακτήρα χωριστά. Το πρόγραμμα διαβάζει το αρχείο και σύμφωνα με το ερωτηματικό (;) διαχωρίζει τα πεδία. Στην περίπτωση που εφαρμόζεται Blocking όπως αναφέρθηκε στην ενότητα 4.2.1, το πρώτο πεδίο, δηλαδή το μπλοκ, κρυπτογραφείται ως έχει και ακολουθεί ο κωδικός της εγγραφής. Για τα υπόλοιπα πεδία, προκειμένου να μετρηθεί η απόσταση σύμφωνα με τον εκάστοτε αλγόριθμο πρέπει να συγκριθούν οι λέξεις χαρακτήρα προς χαρακτήρα. Για το λόγο αυτό κρυπτογραφείται ο κάθε χαρακτήρας ξεχωριστά, χωρίζοντας τους με το θαυμαστικό (!), ενώ τα πεδία μεταξύ τους εξακολουθούν να χωρίζονται με (;).

Κάθε χαρακτήρας ή το πεδίο μπλοκ αποτελεί το κείμενο εισόδου προς κρυπτογράφηση που δίνεται σε μια υπο-ρουτίνα “cryptr” η οποία το παίρνει σαν παράμετρο. Στο σημείο αυτό καλείται η συνάρτηση κρυπτογράφησης md5_hex($ciphertext) του module της MD5. Μετά την εκτέλεση της επιστρέφεται η σύνοψη, που αναφέρεται ως $digest, η οποία γράφεται σε ένα αρχείο κρυπτογραφημένων δεδομένων με τον κωδικό και το πεδίο σε κάθε εγγραφή. Στο παρακάτω παράδειγμα παρουσιάζεται ένα τμήμα του αρχείου για μια εγγραφή, όπου διαφαίνεται ότι έχει εφαρμοστεί Blocking.

ed6966981beb595bd0a50e4371805dec;rec-17; fbade9e36a3f36d3d676c1b808451dd7!0cc175b9c0f1b6a831c399e269772661!7b8b965ad4bca0e41ab51de7b31363a1!e1671797c52e15f763380b45e841ec32!;92eb5ffee6ae2fec3ad71c777531578f!e1671797c52e15f763380b45e841ec32!0cc175b9c0f1b6a831c399e269772661!e358efa489f58062f10dd7316b65649e!e358efa489f58062f10dd7316b65649e!865c0c0b4ab0e063e5caa3387c1a8741!e1671797c52e15f763380b45e841ec32!;

Παρόμοια με τα προηγούμενα προγράμματα, εκτελείται στη συνέχεια το πρόγραμμα "sort_encrypted.plx" όπου γίνεται η ταξινόμηση των δεδομένων.

## 4.3 Προγράμματα Σύγκρισης Ομοιότητας

Τα προγράμματα σύγκρισης της ομοιότητας των δεδομένων είναι αυτά τα οποία, σύμφωνα με τα πρωτόκολλα, χρησιμοποιεί η Κάρολ. Τα προγράμματα αυτά είναι σε θέση να συγκρίνουν τόσο κρυπτογραφημένα όσο και μη κρυπτογραφημένα δεδομένα. Επίσης όσον αφορά το Blocking, τα προγράμματα είναι τα ίδια απλά εφαρμόζουν την σύγκριση μόνο μεταξύ εγγραφών που ανήκουν στο ίδιο μπλοκ. Συνολικά υλοποιήθηκαν τέσσερα προγράμματα:

1. **Πρόγραμμα comp_ngrams**: Δέχεται σαν είσοδο δύο αρχεία με Ν-γράμματα, με όποιον τρόπο και αν έχουν παραχθεί, και κάνει την σύγκριση τους.
2. **Πρόγραμμα comp_trigrams:** Δέχεται σαν είσοδο δύο αρχεία με Τρι-γράμματα και εφαρμόζει την σύγκριση τους.
3. **Πρόγραμμα comp_edit_dist**: Δέχεται σαν είσοδο δύο κατάλληλα επεξεργασμένα αρχεία ώστε να τα συγκρίνει με χρήση της απόστασης Levenshtein.
4. **Πρόγραμμα comp_jaro:** Δέχεται σαν είσοδο δύο κατάλληλα επεξεργασμένα αρχεία ώστε να τα συγκρίνει με χρήση της απόστασης Jaro-Winkler.

### 4.3.1 Πρόγραμμα Σύγκρισης Ν-γραμμάτων

Το πρόγραμμα που επιτυγχάνει την σύγκριση των Ν-γραμμάτων, είτε αυτά έχουν παραχθεί με την μέθοδο δημιουργίας δυναμοσυνόλων είτε με την τροποποιημένη μέθοδο, είναι το "comp_ngrams.plx" που υλοποιήθηκε κατά την διάρκεια προηγούμενης έρευνας [51]. Το πρόγραμμα ξεκινά με το διάβασμα των αρχείων με τους κωδικούς. Ουσιαστικά αυτό που κάνει είναι να εφαρμόζει όλους τους δυνατούς συνδυασμούς κωδικών που παίρνει από τα δύο αρχεία. Έπειτα με οδηγό κάθε ζεύγος κωδικών συγκρίνει τα αρχεία που περιέχουν τα Ν-γράμματα. Συγκεκριμένα παίρνει σαν παραμέτρους στο command line τα δύο αρχεία προς σύγκριση, για παράδειγμα τα "outinput1.csv" και "outinput2.csv", τα δύο αρχεία που περιέχουν τους κωδικούς, το όνομα του αρχείου στο οποίο θα αποθηκευτούν τα αποτελέσματα σαν output, και τον αριθμό των πεδίων κάθε εγγραφής. Η σύγκριση γίνεται κατά α/α πεδίου καταρχάς, και κωδικού του πρώτου αρχείου με όλους τους κωδικούς του δεύτερου αρχείου στη συνέχεια. Για τον λόγο αυτό εφαρμόζεται μία επαναληπτική διαδικασία for από 1 έως n, όπου 'n' είναι ο αριθμός των πεδίων της εγγραφής, στην οποία λαμβάνει χώρα η όλη διαδικασία της σύγκρισης όπως περιγράφεται στη συνέχεια. Προφανώς στην περίπτωση του Blocking η σύγκριση γίνεται με τους κωδικούς που ανήκουν στο ίδιο μπλοκ.

Διαβάζονται τα δύο αρχεία εισόδου που περιέχουν τους κωδικούς, και υλοποιείται μια αλληλουχία του κωδικού και του α/α του πεδίου που υπάρχει από τον επαναληπτικό βρόχο for (1..n). Με βάση αυτό δημιουργείται ένα μοναδικό κλειδί κωδικός - α/α πεδίου για κάθε αρχείο. Στη συνέχεια το πρώτο αρχείο με τους κωδικούς διαβάζεται με μια επαναληπτική διαδικασία while και το δεύτερο όμοια με ένα while εμφωλιασμένο στο πρώτο. Αυτό συμβαίνει έτσι ώστε να επιτευχθούν όλοι οι δυνατοί συνδυασμοί κωδικών.

Κατόπιν χρησιμοποιώντας το παραπάνω κλειδί, γίνεται αναζήτηση των αντίστοιχων εγγραφών στα αρχεία που περιέχουν τα Ν-γράμματα. Εν συνεχεία αυτά διαβάζονται με επαναληπτικό βρόχο while ο οποίος είναι και αυτός εμφωλιασμένος στις παραπάνω επαναληπτικές διαδικασίες. Η σύγκριση γίνεται με τον τρόπο που

περιγράφεται ακολούθως. Υπάρχει ένα κλειδί του πρώτου αρχείου με τους κωδικούς και ένα κλειδί του δεύτερου αρχείου. Εντοπίζονται οι εγγραφές στο πρώτο αρχείο με τα N-γράμματα που το κλειδί τους είναι ίσο με το κλειδί του πρώτου αρχείου με τους κωδικούς. Παρόμοια εντοπίζονται και οι εγγραφές του δεύτερου αρχείου με N-γράμματα σύμφωνα με τα κλειδιά του δεύτερου αρχείου με τους κωδικούς. Στο σημείο αυτό πρέπει να σημειωθεί ότι, όταν διαβάζονται όλες οι παραπάνω εγγραφές κλείνουν τα αρχεία με τα N-γράμματα, και αυτά ανοίγουν πάλι με ένα επόμενο ζεύγος κλειδιών. Δηλαδή αν υφίστανται $N$ πεδία, $K$ εγγραφές κωδικών στο πρώτο αρχείο και $L$ εγγραφές κωδικών στο δεύτερο αρχείο, τότε η προσπέλαση στα δύο αρχεία με τα N-γράμματα γίνεται $N \times K \times L$ φορές.

Στην συνέχεια στα δύο αρχεία που περιέχουν τα N-γράμματα, έχοντας τις εγγραφές ταξινομημένες αλφαριθμητικά, δηλαδή κατά κλειδί και N-γράμμα, υπάρχει η δυνατότητα να γίνει η σύγκριση για ένα ζεύγος κλειδιών κάθε φορά με τον ακόλουθο τρόπο. Αν το N-γράμμα, έστω $A$, του πρώτου αρχείου ταυτίζεται με το N-γράμμα, έστω $B$, του δεύτερου αρχείου, τότε καταμετράται με έναν μετρητή, έστω "cntc". Αν ισχύει "$A < B$" τότε διαβάζεται το πρώτο αρχείο μέχρι να ισχύει "$A \geq B$". Αν ισχύει το ίσο "=" τότε μετράται (cntc++), αλλιώς όταν ισχύει το μεγαλύτερο ">" διαβάζεται το δεύτερο αρχείο. Αυτό συμβαίνει μέχρι αντίστοιχα να βρεθεί "$B \geq A$" οπότε και εφαρμόζεται η ίδια διαδικασία. Δηλαδή διαβάζονται κατά προτεραιότητα τα N-γράμματα που έχουν την μικρότερη τιμή. Επίσης κάθε φορά που διαβάζεται μια εγγραφή, τότε αυτή καταμετράται με έναν μετρητή "cntA" για το πρώτο αρχείο και "cntB" για το δεύτερο. Όταν τελειώσουν οι εγγραφές με το συγκεκριμένο κλειδί στο ένα αρχείο, τότε αυτό κλείνει, αλλά συνεχίζεται η ανάγνωση στο άλλο αρχείο καταμετρώντας τις υπόλοιπες εγγραφές. Αυτό γίνεται μέχρι να βρεθεί ένα άλλο κλειδί, οπότε κλείνει και αυτό και υπολογίζεται το ποσοστό από τον τύπο που δίνεται στην περιγραφή του πρωτοκόλλου.

Δηλαδή θεωρείται ένα πεδίο "n_field" το οποίο ενδέχεται να αντιστοιχεί για παράδειγμα σε ένα επίθετο ή τηλέφωνο. Τότε θα είναι "cntA" τα N-γράμματα του πρώτου αρχείου, έστω με κωδικό $r_i$, και "cntB" τα N-γράμματα του δεύτερου αρχείου, έστω με κωδικό $rec_j$. Αν τα κοινά τους N-γράμματα είναι το "cntc", τότε για κάθε

ζεύγος ($r_i$, $rec_j$) για το πεδίο "n_field", υπολογίζεται το ποσοστό, έστω "ngram_score", σύμφωνα με τον τύπο:

$$\text{ngram\_score} = \frac{\text{cntc}}{0.5 * (\text{cntA} + \text{cntB})}$$

Τέλος γράφονται όλες αυτές οι τιμές για κάθε πεδίο και κάθε ζεύγος κωδικών σε ένα αρχείο αποτελεσμάτων. Επίσης υπάρχει η δυνατότητα, να χρησιμοποιηθεί μια μεταβλητή *m* που να αντιστοιχεί στην τιμή κατωφλίου, έτσι ώστε μόνο οι τιμές με "*ngram_score* ≥ *m*" να εγγράφονται στο αρχείο αποτελεσμάτων. Προφανώς εάν *m = 0* τότε γράφονται όλοι οι δυνατοί συνδυασμοί κωδικών των δύο αρχείων προς σύγκριση και για όλα τα πεδία. Στον πίνακα 4-6 διακρίνεται ένα τμήμα του αρχείου αποτελεσμάτων για την σύγκριση Ν-γραμμάτων με δυναμοσύνολα. Σε αυτό περιέχονται οι κωδικοί των εγγραφών που συγκρίνονται, ο αριθμός των κοινών τους Ν-γραμμάτων, ο αριθμός των Ν-γραμμάτων που έχει κάθε εγγραφή, καθώς και η ποσοστιαία βαθμολογία με βάση τον τύπο.

| **Κωδικός 1ης Εγγραφής** | **Κωδικός 2ης Εγγραφής** | **Κοινά Ν-γράμματα** | **Ν-γράμματα 1ης Εγγραφής** | **Ν-γράμματα 2ης Εγγραφής** | **Ngram score** |
|---|---|---|---|---|---|
| r-2241 | rec-2241 | 127 | 127 | 127 | 1 |
| r-2241 | rec-2561 | 1 | 127 | 127 | 0.007874015748031 |
| r-2241 | rec-2811 | 3 | 127 | 2047 | 0.002759889604416 |
| r-2241 | rec-3761 | 0 | 127 | 15 | 0 |
| r-2561 | rec-2241 | 1 | 127 | 127 | 0.007874015748031 |
| r-2561 | rec-2561 | 127 | 127 | 127 | 1 |
| r-2561 | rec-2811 | 1 | 127 | 2047 | 0.000919963201472 |
| r-2561 | rec-3761 | 3 | 127 | 15 | 0.042253521126761 |
| r-2811 | rec-2241 | 3 | 2047 | 127 | 0.002759889604416 |
| r-2811 | rec-2561 | 1 | 2047 | 127 | 0.000919963201472 |
| r-2811 | rec-2811 | 1023 | 2047 | 2047 | 0.499755740107474 |
| r-2811 | rec-3761 | 0 | 2047 | 15 | 0 |

Πίνακας 4-6: Παράδειγμα αρχείου αποτελεσμάτων για Ν-γράμματα

### 4.3.2 Πρόγραμμα Σύγκρισης Τρι-γραμμάτων

Η διαδικασία της σύγκρισης Τρι-γραμμάτων επιτυγχάνεται με το πρόγραμμα "comp_trigrams.plx". Το διάβασμα των αρχείων και η διαχείριση των εγγραφών και των κωδικών τους γίνεται ακριβώς με τον ίδιο τρόπο που περιγράφθηκε στην ενότητα 4.3.1. Αφού εκτελεστούν αυτές οι διαδικασίες, η σύγκριση για ένα ζεύγος κλειδιών επιτυγχάνεται με τον ακόλουθο τρόπο. Αν το Τρι-γράμμα, έστω *A*, του πρώτου αρχείου ταυτίζεται με το Τρι-γράμμα, έστω *B*, του δεύτερου αρχείου, τότε καταμετράται με έναν μετρητή, έστω "cntc". Αντίθετα, αν το *A* εμφανίζεται "tgramA" φορές και το *B* "tgramB" φορές, τότε υπολογίζεται το τετράγωνο της διαφοράς τους, δηλαδή το $(tgramA - tgramB)^2$. Αυτό αθροίζεται σε ένα αθροιστή, έστω "sum", ο οποίος στην αρχή θα είναι ίσος με το μηδέν. Αν ισχύει "$A < B$" τότε διαβάζεται το πρώτο αρχείο μέχρι να γίνει "$A \geq B$", υπολογίζοντας κάθε φορά το τετράγωνο του "tgramA" αθροιζόμενο στον αθροιστή "sum". Όταν ισχύει "$A = B$" τότε μετράται "cntc++" και υπολογίζεται πάλι το τετράγωνο της διαφοράς τους, αθροιζόμενο στον αθροιστή "sum". Από την άλλη, όταν ισχύει "$A > B$" διαβάζεται το δεύτερο αρχείο. Αυτό συμβαίνει μέχρι αντίστοιχα να βρεθεί "$B \geq A$" οπότε και εφαρμόζεται η ίδια διαδικασία υπολογίζοντας αυτή τη φορά το τετράγωνο του "tgramB" αθροιζόμενο στον ίδιο αθροιστή "sum". Δηλαδή διαβάζονται κατά προτεραιότητα τα Τρι-γράμματα που έχουν την μικρότερη τιμή. Επίσης κάθε φορά που διαβάζεται μια εγγραφή, τότε αυτή καταμετράται με έναν μετρητή "cntA" για το πρώτο αρχείο και "cntB" για το δεύτερο. Όταν τελειώσουν οι εγγραφές με το συγκεκριμένο κλειδί στο ένα αρχείο, τότε αυτό κλείνει, αλλά συνεχίζεται η ανάγνωση στο άλλο αρχείο καταμετρώντας τις υπόλοιπες εγγραφές και αθροίζοντας τα αντίστοιχα τετράγωνα του αριθμού εμφανίσεως τους. Αυτό γίνεται μέχρι να βρεθεί ένα άλλο κλειδί, οπότε κλείνει και αυτό και υπολογίζεται η ένταση της διαφοράς των διανυσμάτων, δηλαδή η τετραγωνική ρίζα του "sum", όπως φαίνεται από τον τύπο που δόθηκε στην ενότητα 3.3:

$$\left\|\vec{D}\right\| = \sqrt{\sum_{i=aaa}^{999} (a_i - b_i)^2}$$

όπου $a_i$, $b_i$ είναι ο αριθμός εμφάνισης των Τρι-γραμμάτων.

Κατόπιν αυτό συγκρίνεται, σύμφωνα με τον τύπο που δίνεται στην περιγραφή του πρωτοκόλλου, με την τιμή κατωφλίου "$T = 2.486 + 0.025n$", όπου $n$ είναι ο συνολικός αριθμός των διακριτών Τρι-γραμμάτων των πεδίων προς σύγκριση.

Για να γίνει αυτό κατανοητό δίνεται το εξής παράδειγμα. Έστω το πεδίο $A$ είναι "801150115011", οπότε παράγεται το ακόλουθο διάνυσμα Τρι-γραμμάτων:

$$(011(3), 115(2), 150(2), 501(2), 801(1)),$$

όπου εντός παρενθέσεων είναι ο αριθμός εμφάνισης τους.

Παρόμοια αν το πεδίο $B$ είναι "9011501150" παράγεται το διάνυσμα Τρι-γραμμάτων:

$$(011(2), 115(2), 150(2), 501(1), 901(1)).$$

Το διάνυσμα της διαφοράς των $A$ και $B$ θα είναι

$$(011(3 - 2), 115(2 - 2), 150(2 - 2), 501(2 - 1), 801(1 - 0), 901(1 - 0)) =$$
$$= (011(1), 501(1), 801(1), 901(1)),$$

δηλαδή η διαφορά των αριθμών εμφάνισης των Τρι-γραμμάτων. Επομένως, η ένταση του διανύσματος της διαφοράς θα είναι

$$\|\vec{D}\| = \sqrt{1^2 + 1^2 + 1^2 + 1^2} = 2.$$

Στο παράδειγμά ισχύει $n = 6$, άρα $T = 2.486 + 0.025 \times 6 = 2,636$. Τελικά παρατηρείται ότι ισχύει $2 \leq 2,636 \leftrightarrow \|\vec{D}\| \leq T$, άρα τα πεδία ταιριάζουν.

Γενικεύοντας, για ένα πεδίο "n_field", θα είναι "cntA" τα Τρι-γράμματα του πρώτου αρχείου, έστω με κωδικό $r_i$, και "cntB" τα Τρι-γράμματα του δεύτερου αρχείου, έστω με κωδικό $rec_j$. Αν τα κοινά τους Τρι-γράμματα είναι το "cntc", τότε για κάθε ζεύγος ($r_i$, $rec_j$), υπολογίζεται το κατώφλι και η ένταση του διανύσματος διαφοράς.

Εφόσον ισχύει $\|\vec{D}\| \leq T$ γράφονται όλες αυτές οι τιμές για κάθε πεδίο και κάθε ζεύγος κωδικών σε ένα αρχείο αποτελεσμάτων. Στον πίνακα 4-7 διακρίνεται ένα τμήμα του αρχείου αποτελεσμάτων με τις τιμές που ικανοποιούν την παραπάνω συνθήκη. Σε αυτό περιέχονται οι κωδικοί των εγγραφών που συγκρίνονται και ικανοποιούν την συνθήκη, η ένταση του διανύσματος της διαφοράς $\|\vec{D}\|$ και το κατώφλι $T$.

| Κωδικός 1ης Εγγραφής | Κωδικός 2ης Εγγραφής | Ένταση διαφοράς | Κατώφλι |
|---|---|---|---|
| r-2241 | rec-2241 | 1.4142135623731 | 2.661 |
| r-2561 | rec-2561 | 0 | 2.636 |
| r-3761 | rec-3761 | 2.23606797749979 | 2.711 |
| r-2242 | rec-2242 | 1.4142135623731 | 2.611 |
| r-2242 | rec-3762 | 2.64575131106459 | 2.661 |

Πίνακας 4-7: Παράδειγμα αρχείου αποτελεσμάτων για Τρι-γράμματα

### 4.3.3 Πρόγραμμα Σύγκρισης με Απόσταση Levenshtein

Η σύγκριση της ομοιότητας των εγγραφών με την απόσταση Levenshtein επιτυγχάνεται με το πρόγραμμα "comp_edit_dist.plx". Το πρόγραμμα ξεκινά με το διάβασμα των αρχείων που έχουν παραχθεί από την διαδικασία της κρυπτογράφησης. Διαβάζει την πρώτη εγγραφή του πρώτου αρχείου και την συγκρίνει με όλες τις εγγραφές του δεύτερου. Κατόπιν διαβάζει την δεύτερη εγγραφή και την συγκρίνει με όλες επίσης τις εγγραφές του δεύτερου αρχείου και συνεχίζει έτσι μέχρι να διαβάσει όλες τις εγγραφές του πρώτου. Ουσιαστικά αυτό που κάνει είναι να εφαρμόζει όλους τους δυνατούς συνδυασμούς των εγγραφών που παίρνει απ' τα δύο αρχεία, όπως κάνουν και τα δύο προηγούμενα προγράμματα που περιγράφθηκαν.

Στη συνέχεια, με βάση το ερωτηματικό (;) διαχωρίζει τα κρυπτογραφημένα πεδία της κάθε εγγραφής. Έπειτα υπολογίζει την απόσταση των αντίστοιχων πεδίων που είναι για σύγκριση με μια μέθοδο που βασίζεται στον αλγόριθμο της απόστασης Levenshtein. Όπως προαναφέρθηκε στην ενότητα 4.2.2 το κάθε πεδίο έχει κρυπτογραφηθεί χαρακτήρα προς χαρακτήρα, όπου μεταξύ τους είναι διαχωρισμένοι με το θαυμαστικό (!). Με βάση αυτό οι χαρακτήρες των δύο πεδίων αποθηκεύονται σε δύο αντίστοιχους πίνακες που είναι παράμετροι της υπο-ρουτίνας "Ldist".

Η υπο-ρουτίνα "Ldist" υπολογίζει την απόσταση Levenstein των δύο αλφαριθμητικών ως εξής. Καταρχάς αν το ένα αλφαριθμητικό είναι το κενό η

απόσταση είναι το μήκος του άλλου. Αν το ένα αλφαριθμητικό έχει μήκος "$len_1$" και το άλλο έχει μήκος "$len_2$", τότε ορίζεται ένας δισδιάστατος πίνακας "mat" ο οποίος έχει δομή πίνακα κατακερματισμού. Ο πίνακας αυτός θα έχει "$len_1+1$" στήλες και "$len_2+1$" γραμμές, και αρχικοποιείται θέτοντας στην πρώτη γραμμή τους αριθμούς "$0,1,....len_1$", στην πρώτη στήλη "$0,1,....len_2$", και σε όλα τα υπόλοιπα το μηδέν. Εφαρμόζοντας μία επαναληπτική διαδικασία for από *1* έως $len_1$ και ένα εμφωλιασμένο for από *1* έως $len_2$, υπολογίζεται η απόσταση, συγκρίνοντας κάθε χαρακτήρα του πρώτου αλφαριθμητικού με όλους τους χαρακτήρες του δεύτερου, σύμφωνα με τον τρόπο που περιγράφθηκε στην ενότητα 2.1.2. Κάθε φορά το κόστος θα αποθηκεύεται σε μια μεταβλητή "cost" η οποία θα παίρνει τιμές μηδέν ή ένα, ανάλογα αν οι χαρακτήρες είναι ίσοι ή όχι. Αντίστοιχα, θεωρώντας "*d(i,j)*" το πλήθος των πράξεων, στο (*i-στο,j-στο*) στοιχείο του πίνακα θα ανατίθεται η τιμή *"min(d(i-1, j) + 1, d(i, j-1) + 1, d(i-1, j-1) + cost)"*. Όταν γεμίσει ο πίνακας θα επιστραφεί η τιμή του τελευταίου στοιχείου ($len_1$,$len_2$) μέσω της μεταβλητής "dist", η οποία είναι η απόσταση των δύο αλφαριθμητικών. Στην συνέχεια το κλάσμα της απόστασης δια του αθροίσματος των μηκών των αλφαριθμητικών, αποθηκεύεται στην μεταβλητή "$fraction = dist/(len_1+len_2)$".

Τέλος γράφονται οι τιμές της απόστασης και του κλάσματος για κάθε πεδίο και κάθε ζεύγος κωδικών σε ένα αρχείο αποτελεσμάτων. Επίσης υπάρχει η δυνατότητα, να χρησιμοποιηθεί μια μεταβλητή "m" που να αντιστοιχεί στην τιμή κατωφλίου, έτσι ώστε μόνο οι τιμές με "$fraction \leq m$" να εγγράφονται στο αρχείο αποτελεσμάτων. Πρέπει να αναφερθεί, ότι αν εφαρμόζεται Blocking, προφανώς συγκρίνονται μόνο εγγραφές που ανήκουν στο ίδιο μπλοκ. Στον πίνακα 4-8 διακρίνεται ένα τμήμα του αρχείου αποτελεσμάτων της απόστασης Levenshtein.

| Κωδικός 1ης Εγγραφής | Κωδικός 2ης Εγγραφής | Αριθμός Αντιμεταθέσεων | Κατώφλι |
|---|---|---|---|
| r-2091 | rec-2704 | 1 | 0.0526 |
| r-2932 | rec-2571 | 1 | 0.0909 |
| r-3461 | rec-2422 | 1 | 0.0714 |
| r-3911 | rec-3473 | 2 | 0.125 |
| r-3452 | rec-3110 | 2 | 0.1818 |

Πίνακας 4-8: Παράδειγμα αρχείου αποτελεσμάτων για την απόσταση Levenshtein

### 4.3.4 Πρόγραμμα Σύγκρισης με Απόσταση Jaro-Winkler

Το πρόγραμμα "comp_jaro.plx" υπολογίζει την απόσταση των αλφαριθμητικών με χρήση του αλγόριθμου Jaro – Winkler. Η υπο-ρουτίνα "Jaro_Winkler" παίρνει ως παραμέτρους τα κρυπτογραφημένα πεδία που θα συγκριθούν και όχι τους χαρακτήρες των πεδίων ένα προς ένα όπως στην περίπτωση της απόστασης Levenshtein. Οι χαρακτήρες πάλι είναι κρυπτογραφημένοι και χωρίζονται με θαυμαστικό (!), αλλά ο διαχωρισμός τους γίνεται μέσα στην υπο-ρουτίνα με την βοήθεια της εντολής "@char1=split(/!/,$s1);".

Οι πίνακες "$char_1$" και "$char_2$" περιέχουν τους κρυπτογραφημένους χαρακτήρες των αντίστοιχων αλφαριθμητικών $s_1, s_2$ που είναι προς σύγκριση. Μάλιστα κάθε χαρακτήρας αποτελεί και ένα στοιχείο του πίνακα. Αρχικά υπολογίζεται η μέγιστη απόσταση που επιτρέπεται να έχουν δύο χαρακτήρες ώστε να θεωρηθούν κοινοί, και αποθηκεύεται στην μεταβλητή "maxs1s2". Κατόπιν εφαρμόζεται μία επαναληπτική διαδικασία for με δύο μεταβλητές "i" και "j", η πρώτη για το βηματισμό του πρώτου αλφαριθμητικού και η δεύτερη για το δεύτερο, μέχρι να φτάσει στο τέλος ενός εκ των δύο. Σε αυτήν εξετάζεται αν οι αντίστοιχοι χαρακτήρες στην ίδια θέση, δηλαδή "$i = j$" είναι ίσοι, με χρήση της εντολής "$char_1[$i] eq $char_2[$j]". Αν είναι ίσοι, ο κοινός χαρακτήρας ανατίθεται στον πίνακα "$match_1$", ενώ στον πίνακα "$match_2$" στην

αντίστοιχη θέση ανατίθεται το μηδέν για να μην εξεταστεί ξανά παρακάτω. Δηλαδή στην περίπτωση όπου οι χαρακτήρες δεν ίσοι, εφαρμόζεται εντός του πρώτου αλφαριθμητικού μία επαναληπτική διαδικασία for με μεταβλητή "$j_1$", η οποία ψάχνει την ισότητα με εύρος "$i - maxs1s2 \leq j_1 \leq i + maxs1s2$", δηλαδή όσο είναι το εύρος των κοινών χαρακτήρων. Αν ήδη ισχύει "$match_2[j_1] = 0$" παραλείπεται η διαδικασία, αλλιώς ο κοινός χαρακτήρας ανατίθεται στον πίνακα "$match_1$". Κάθε φορά που υπάρχει μια ισότητα χαρακτήρων, αυξάνεται μια μεταβλητή "smatch1" κατά μια μονάδα. Αυτή στο τέλος περιέχει τον αριθμό των κοινών χαρακτήρων των δύο αλφαριθμητικών.

Έπειτα με μία επαναληπτική διαδικασία for μετράται ο αριθμός των αντιμεταθέσεων των κοινών χαρακτήρων που βρέθηκαν ίσοι και δεν ήταν στην αντίστοιχη θέση, και αποθηκεύεται στην μεταβλητή "transp1". Επίσης στην αρχική επαναληπτική διαδικασία υπολογίζεται το κοινό πρόθεμα των δύο αλφαριθμητικών, το οποίο αποθηκεύεται στην μεταβλητή "prefix".

Τέλος στην μεταβλητή "Jaro" αποθηκεύεται το αποτέλεσμα της εφαρμογής του κανόνα του Jaro όπου σύμφωνα με τον τύπο της ενότητας 2.1.3, θα είναι

$$Jaro = \frac{1}{3}\left(\frac{smatch\ 1}{len1} + \frac{smatch\ 1}{len2} + \frac{smatch\ 1 - transp\ 1}{smatch\ 1}\right),$$

ενώ στην μεταβλητή "Jaro_Winkler" αποθηκεύεται το αποτέλεσμα της εφαρμογής του αλγόριθμου Jaro-Winkler, όπου θα είναι:

$$Jaro\_Winkler = Jaro + P * \cdot 0.1 * (1 - Jaro) \text{ αν } smatch1 > 0,$$

$$Jaro\_Winkler = 0 \text{ αν } smatch1 = 0,$$

όπου το "$P$" θα ισούται με το πρόθεμα αν αυτό είναι μικρότερο του 4, ενώ σε κάθε άλλη περίπτωση θα ισούται με 4.

Η υπο-ρουτίνα "Jaro_Winkler" επιστρέφει την τιμή της μεταβλητής "Jaro_Winkler" στο κυρίως πρόγραμμα όπου γράφονται οι τιμές της απόστασης για κάθε πεδίο και κάθε ζεύγος κωδικών σε ένα αρχείο αποτελεσμάτων. Επίσης υπάρχει η δυνατότητα, να χρησιμοποιηθεί μια μεταβλητή "m" που να αντιστοιχεί στην τιμή κατωφλίου, έτσι ώστε μόνο οι τιμές με "$Jaro\_Winkler > m$" να εγγράφονται στο αρχείο

αποτελεσμάτων. Στον πίνακα 4-9 διακρίνεται ένα τμήμα του αρχείου αποτελεσμάτων της απόστασης Jaro – Winkler.

| **Κωδικός 1ης Εγγραφής** | **Κωδικός 2ης Εγγραφής** | **Κατώφλι** |
|---|---|---|
| r-2851 | rec-2988 | 0.98 |
| r-3794 | rec-2003 | 0.9666 |
| r-2456 | rec-3158 | 0.9428 |
| r-3324 | rec-3011 | 0.9416 |
| r-2152 | rec-2693 | 0.84 |

Πίνακας 4-9: Παράδειγμα αρχείου αποτελεσμάτων για την απόσταση Jaro - Winkler

# Κεφάλαιο 5

# Αξιολόγηση Πρωτοκόλλων

Στο παρόν κεφάλαιο θα γίνει η αξιολόγηση όλων των πρωτοκόλλων που υλοποιήθηκαν κατά την διάρκεια της έρευνας. Ο ένας στόχος της αξιολόγησης είναι να εξετάσει τον χρόνο που απαιτείται προκειμένου να εκτελεστεί κάθε πρωτόκολλο, ανάλογα με τον όγκο των δεδομένων που συγκρίνονται. Ο δεύτερος στόχος είναι να ελέγξει κατά πόσο κάθε πρωτόκολλο επιτυγχάνει ορθά την διαδικασία της διασύνδεσης γνωρισμάτων.

Στην κατεύθυνση αυτή εκτελέστηκαν αρκετά πειράματα. Στα πειράματα αυτά χρησιμοποιήθηκαν διάφορα μεγέθη συνόλων δεδομένων, από πολύ μικρά ως αρκετά μεγάλα, ώστε κάποιος να έχει στην διάθεση του μια γενική εικόνα των χρονικών επιδόσεων των πρωτοκόλλων. Παράλληλα εξετάστηκαν αλφαριθμητικά διαφόρων μηκών και με αρκετές διαφοροποιήσεις, ώστε να γίνει σωστή αποτίμηση της ικανότητας των πρωτοκόλλων στο ταίριασμα των γνωρισμάτων, και κατά επέκταση, των εγγραφών.

Τα σύνολα δεδομένων που χρησιμοποιήθηκαν κατά την διάρκεια των πειραμάτων δημιουργήθηκαν με την χρήση του λογισμικού Febrl [9], με τον τρόπο που περιγράφεται στο Παράρτημα της εργασίας. Κάθε ένα από αυτά τα σύνολα δεδομένων αποτελείται από 14 πεδία, εκ των οποίων το πρώτο είναι ο αναγνωριστικός κωδικός κάθε εγγραφής, ενώ τα υπόλοιπα είναι μεταξύ άλλων το ονοματεπώνυμο, η διεύθυνση και ο αριθμός ασφάλισης. Επίσης αξίζει να σημειωθεί ότι τα πειράματα εκτελέστηκαν σε έναν προσωπικό υπολογιστή με επεξεργαστή Intel Core 2 Duo στα 3.0 GHz, με μνήμη RAM 2 Gigabyte και λειτουργικό σύστημα Suse Linux 8.1.

## 5.1 Αξιολόγηση Χρονικών Αποτελεσμάτων

Σε αυτήν την ενότητα θα γίνει η παρουσίαση της χρονικής διάρκειας που χρειάζεται για να εκτελεστεί κάθε πρωτόκολλο. Προκειμένου να επιτευχθεί αυτός ο στόχος χρησιμοποιήθηκαν διάφορα ζεύγη συνόλων δεδομένων, τα οποία περιελάμβαναν από 4 ως αρκετές χιλιάδες εγγραφές. Τα μικρά σύνολα δεδομένων χρησιμοποιήθηκαν για την εξέταση των πρωτοκόλλων που είχαν μεγάλες χρονικές απαιτήσεις, όπως είναι το πρωτόκολλο χρήσης Ν-γραμμάτων. Από την άλλη πλευρά, τα μεγαλύτερα σύνολα δεδομένων χρησιμοποιήθηκαν για την εξέταση των σχετικά γρηγορότερων πρωτοκόλλων, όπως είναι αυτά που κάνουν χρήση των μετρικών απόστασης.

Μια γενικότερη διαπίστωση η οποία θα φανεί και στην συνέχεια της ενότητας είναι ότι η διαδικασία της κρυπτογράφησης επιτυγχάνεται σε πολύ μικρό χρονικό διάστημα, ανεξαρτήτως του πρωτοκόλλου και του μεγέθους των συνόλων δεδομένων.

### 5.1.1 Πρωτόκολλο Χρήσης Ν-γραμμάτων

Τα πειραματικά αποτελέσματα, όσον αφορά τους χρόνους, για το πρωτόκολλο χρήσης Ν-γραμμάτων ήταν αρκετά απογοητευτικά. Το πρόγραμμα δημιουργίας Ν-γραμμάτων αντιμετώπισε ένα ιδιαίτερα σημαντικό πρόβλημα. Η γλώσσα προγραμματισμού Perl, κατά την δημιουργία των Ν-γραμμάτων, δεν έχει την δυνατότητα να τα αποθηκεύσει ένα-ένα στο δίσκο, αλλά όλα μαζί. Αυτό έχει σαν αποτέλεσμα κατά την επεξεργασία εκατομμυρίων Ν-γραμμάτων να παρουσιάζεται πρόβλημα υπερχείλισης μνήμης. Συγκεκριμένα η Perl, ανεξαρτήτως του υπολογιστικού συστήματος, μπορεί να χρησιμοποιήσει μέχρι περίπου 261 Mbytes μνήμης RAM. Επομένως κατά την επεξεργασία ενός μεγάλου πλήθους Ν-γραμμάτων, όπου χρειάζεται περισσότερη μνήμη, δημιουργείται σφάλμα παραβίασης του μέγιστου επιτρεπτού ορίου χρήσης μνήμης.

Όπως φαίνεται και στον πίνακα 5-1 που παρουσιάζει στοιχεία για Δι-γράμματα, σε σύνολα δεδομένων με 80 εγγραφές και άνω δεν κατέστη δυνατή η παραγωγή των Ν-

γραμμάτων. Αυτό μάλιστα ισχύει για οποιαδήποτε τιμή του *N*. Όσον αφορά τα δεδομένα που ήταν δυνατή η επεξεργασία τους, ο χρόνος παραγωγής των N-γραμμάτων αυξάνει όσο αυξάνεται και το πλήθος τους. Επίσης ο χρόνος εκτέλεσης του προγράμματος σύγκρισης της ομοιότητας των N-γραμμάτων είναι ιδιαίτερα μεγάλος. Μάλιστα η διάρκεια εκτέλεσης αυξάνεται εκθετικά καθώς αυξάνεται το πλήθος των προς σύγκριση N-γραμμάτων. Αυτό συμβαίνει διότι κάθε πεδίο κάθε εγγραφής ξεχωριστά πρέπει να συγκριθεί με τα αντίστοιχα πεδία όλων των εγγραφών του άλλου συνόλου δεδομένων.

| **Πλήθος Εγγραφών Συνόλου** | **Πλήθος δι-γραμμάτων** | **Χρόνος σε sec δημιουργίας δι-γραμμάτων** | **Χρόνος σε sec κρυπτογρά-φησης** | **Χρόνος σε sec σύγκρισης** |
|---|---|---|---|---|
| 4 | 20,674 | 2 | 1 | 78 |
| 4 | 37,104 | 3 | 2 | 78 |
| 8 | 34,738 | 2 | 2 | 662 |
| 8 | 30,468 | 2 | 2 | 662 |
| 20 | 81,284 | 5 | 3 | 1.23 ώρες |
| 20 | 145,654 | 10 | 6 | 1.23 ώρες |
| 40 | 1,133,824 | 97 | 37 | 5.36 ώρες |
| 40 | 797,186 | 41 | 30 | 5.36 ώρες |
| 80 | Λάθος μνήμης | ---------- | ---------- | ---------- |

Πίνακας 5-1: Στατιστικά Στοιχεία για τα Δι-γράμματα

Παράλληλα, για τα σύνολα με τις 40 εγγραφές εξετάστηκαν και περιπτώσεις επεξεργασίας N-γραμμάτων με $N > 2$. Συγκεκριμένα έγινε παραγωγή Τρι-γραμμάτων, Τετρα-γραμμάτων, Πεντα-γραμμάτων και Εξι-γραμμάτων. Όπως είναι λογικό καθώς αυξάνεται το μήκος των N-γραμμάτων μειώνεται αισθητά το πλήθος τους που παράγεται, καθώς και ο χρόνος επεξεργασίας τους. Παρά όλα αυτά το συνολικό πλήθος των N-γραμμάτων προς σύγκριση είναι αρκετά μεγάλο, οπότε ο χρόνος που διαρκεί η σύγκριση και πάλι απογοητευτικός. Τα παραπάνω στοιχεία παρουσιάζονται στον πίνακα 5-2.

| Πλήθος Εγγραφών Συνόλου | Μήκος Ν-γραμμάτων | Πλήθος Ν-γραμμάτων | Χρόνος σε sec δημιουργίας Ν-γραμμάτων |
|---|---|---|---|
| 40 | 2 | 1,133,824 | 97 |
| 40 | 3 | 707,520 | 59 |
| 40 | 4 | 394,972 | 31 |
| 40 | 5 | 230,242 | 17 |
| 40 | 6 | 115,130 | 8 |

Πίνακας 5-2: Στατιστικά Στοιχεία Διαφορετικού Μήκους Ν-γραμμάτων του Ίδιου Συνόλου Εγγραφών

Η εφαρμογή του Blocking δεν επέφερε αρκετά πιο ικανοποιητικά αποτελέσματα για την διάρκεια εκτέλεσης του πρωτοκόλλου. Μάλιστα τα αποτελέσματα που εξάγονται δεν είναι αντικειμενικά αφού μπορεί να εφαρμοστεί μόνο σε σύνολα δεδομένων με 40 εγγραφές. Αυτό έχει ως συνέπεια οι εγγραφές που θα ανήκουν σε κοινά μπλοκ να είναι ιδιαίτερα λίγες.

### 5.1.2 Τροποποιημένο Πρωτόκολλο Χρήσης Ν-γραμμάτων

Με την υλοποίηση του τροποποιημένου πρωτοκόλλου χρήσης Ν-γραμμάτων ξεπεράστηκαν τα προβλήματα μνήμης που εμφανίστηκαν στο προηγούμενο πρωτόκολλο. Πλέον το πρόγραμμα προετοιμασίας των δεδομένων είναι σε θέση να παραγάγει Ν-γράμματα από σύνολα δεδομένων οποιουδήποτε μεγέθους. Συγκεκριμένα ήταν δυνατή η επεξεργασία ενός συνόλου δεδομένων που περιείχε 100.000 εγγραφές.

Παράλληλα, το πλήθος των παραγόμενων Ν-γραμμάτων είναι κατά πολύ μικρότερο από αυτό του αρχικού πρωτοκόλλου. Φυσικά αυτό οφείλεται στο γεγονός ότι πλέον η διαδικασία εφαρμόζεται μόνο σε δύο από τα δεκατρία πεδία κάθε εγγραφής και παράγονται μόνο οι δύο υπο-λίστες Ν-γραμμάτων.

Από την άλλη πλευρά, το αρνητικό σημείο της υλοποίησης είναι η διαδικασία της σύγκρισης. Παρά τις βελτιώσεις που υπέστη το πρωτόκολλο, η σύγκριση της

ομοιότητας των γνωρισμάτων είναι ιδιαίτερα χρονοβόρα. Αυτό καθιστά το πρωτόκολλο μη λειτουργικό για την επεξεργασία μεγάλων συνόλων δεδομένων. Μάλιστα επειδή το υπολογιστικό σύστημα στο οποίο εκτελέστηκαν τα προγράμματα είχε μικρές υπολογιστικές δυνατότητες, δεν ήταν δυνατή χρονικά η εκτέλεση του προγράμματος σύγκρισης για μεγάλα σύνολα δεδομένων. Για τον λόγο αυτό στις περιπτώσεις επεξεργασίας 5.000 εγγραφών και άνω, υπολογίστηκαν κατά προσέγγιση οι χρόνοι εκτέλεσης της σύγκρισης, με βάση τους χρόνους που χρειάστηκαν λίγες εγγραφές από το ένα σύνολο δεδομένων να συγκριθούν με όλες τις εγγραφές του άλλου συνόλου δεδομένων. Όλα τα παραπάνω στοιχεία παρουσιάζονται στον πίνακα 5-3, ο οποίος αφορά την επεξεργασία Δι-γραμμάτων.

| **Πλήθος Εγγραφών Συνόλων** | **Πλήθος δι-γραμμάτων** | **Χρόνος σε sec δημιουργίας δι-γραμμάτων** | **Χρόνος σε sec κρυπτογράφησης** | **Χρόνος σε sec σύγκρισης** |
|---|---|---|---|---|
| 40×40 | 1,671× 1,389 | 1 | 1 | 42 |
| 100×100 | 3,186×3,256 | 1 | 1 | 470 |
| 200×200 | 6,766×6,575 | 1 | 1 | 3,543 |
| 300×300 | 10,679×10,561 | 1 | 1 | 3.67 ώρες |
| 500×500 | 17,065×17,726 | 1 | 1 | 8.33 ώρες |
| 1,000×1,000 | 34,492× 36,060 | 1 | 1 | 66.54 ώρες |
| 5,000×1,000 | 167,467×34,492 | 2 | 1 | ≈1,111.12 ώρες |
| 5,000×5,000 | 167,467×173,859 | 3 | 1 | ≈9,384.45 ώρες |
| 10,000×10,000 | 338,617×327,845 | 6 | 1 | ≈66,754.67 ώρες |

Πίνακας 5-3: Στατιστικά Στοιχεία Τροποποιημένου Πρωτοκόλλου για τα Δι-γράμματα

Επίσης για τα σύνολα δεδομένων με τις 1.000 εγγραφές εξετάστηκαν οι χρόνοι δημιουργίας και σύγκρισης Τρι-γραμμάτων, Τετρα-γραμμάτων, Πεντα-γραμμάτων και Εξι-γραμμάτων. Τα στοιχεία αυτά φαίνονται στον πίνακα 5-4.

| Πλήθος Εγγραφών Συνόλων | Μήκος N-γραμμάτων | Πλήθος N-γραμμάτων | Χρόνος σε sec δημιουργίας N-γραμμάτων | Χρόνος σε sec σύγκρισης |
|---|---|---|---|---|
| 1,000×1,000 | 2 | 36,060×34,492 | 1 | 66.54 ώρες |
| 1,000×1,000 | 3 | 26,284×24,578 | 1 | 39.02 ώρες |
| 1,000×1,000 | 4 | 18,184×16,326 | 1 | 31.67 ώρες |
| 1,000×1,000 | 5 | 12,226×10,196 | 1 | 17.78 ώρες |
| 1,000×1,000 | 6 | 8,330×6,220 | 1 | 13.89 ώρες |

Πίνακας 5-4: Στατιστικά Στοιχεία Διαφορετικού Μήκους N-γραμμάτων των Ίδιων Συνόλων Εγγραφών

Με τα ίδια σύνολα δεδομένων εκτελέστηκαν πειράματα στα οποία γινόταν εφαρμογή της διαδικασίας του Blocking. Τα αποτελέσματα αυτών των πειραμάτων διαφαίνονται στους πίνακες 5-5 και 5-6.

| Πλήθος Εγγραφών Συνόλων | Πλήθος δι-γραμμάτων | Χρόνος σε sec δημιουργίας δι-γραμμάτων | Χρόνος σε sec κρυπτογρά-φησης | Χρόνος σε sec σύγκρισης |
|---|---|---|---|---|
| 40×40 | 1,671× 1,389 | 1 | 1 | 1 |
| 100×100 | 3,186×3,256 | 1 | 1 | 3 |
| 200×200 | 6,766×6,575 | 1 | 1 | 16 |
| 300×300 | 10,679×10,561 | 1 | 1 | 39 |
| 500×500 | 17,065×17,726 | 1 | 1 | 178 |
| 1,000×1,000 | 34,492× 36,060 | 1 | 1 | 20.05 λεπτά |
| 5,000×1,000 | 167,467×34,492 | 2 | 1 | 59.5 λεπτά |
| 5,000×5,000 | 167,467×173,859 | 3 | 1 | 502.06 λεπτά |
| 10,000×10,000 | 338,617×327,845 | 6 | 1 | 41.6 ώρες |

Πίνακας 5-5: Στατιστικά Στοιχεία Τροποποιημένου Πρωτοκόλλου για τα Δι-γράμματα με Blocking

| Πλήθος Εγγραφών Συνόλων | Μήκος N-γραμμάτων | Πλήθος N-γραμμάτων | Χρόνος σε sec δημιουργίας N-γραμμάτων | Χρόνος σε min σύγκρισης |
|---|---|---|---|---|
| 1,000×1,000 | 2 | 36,060×34,492 | 1 | 20.05 |
| 1,000×1,000 | 3 | 26,284×24,578 | 1 | 7.31 |
| 1,000×1,000 | 4 | 18,184×16,326 | 1 | 5.05 |
| 1,000×1,000 | 5 | 12,226×10,196 | 1 | 3.26 |
| 1,000×1,000 | 6 | 8,330×6,220 | 1 | 2.05 |

Πίνακας 5-6: Στατιστικά Στοιχεία Διαφορετικού Μήκους N-γραμμάτων των Ίδιων Συνόλων Εγγραφών με Blocking

### 5.1.3 Πρωτόκολλο Χρήσης Τρι-γραμμάτων

Τα αποτελέσματα των πειραμάτων όσον αφορά την διάρκεια εκτέλεσης του πρωτοκόλλου χρήσης Τρι-γραμμάτων δεν είναι ιδιαίτερα ικανοποιητικά. Ενώ τα προγράμματα προετοιμασίας και κρυπτογράφησης των δεδομένων ήταν ιδιαίτερα γρήγορα, το πρόγραμμα σύγκρισης ήταν πολύ αργό. Όπως και στην περίπτωση της ενότητας 5.1.2, οι χρόνοι σύγκρισης συνόλων δεδομένων με 5.000 εγγραφές υπολογίστηκαν προσεγγιστικά, αφού δεν ήταν χρονικά εφικτή η εκτέλεση του προγράμματος.

Για τον λόγο αυτό υλοποιήθηκε μια έκδοση του πρωτοκόλλου στην οποία η επεξεργασία και σύγκριση των δεδομένων εφαρμόζεται μόνο σε δύο πεδία κάθε εγγραφής. Με την μέθοδο αυτή, η οποία εφαρμόστηκε και στο τροποποιημένο πρώτο πρωτόκολλο, επιτεύχθηκε μεγάλη βελτίωση των χρόνων εκτέλεσης του προγράμματος σύγκρισης των γνωρισμάτων.

Από την άλλη πλευρά, η εφαρμογή του Blocking επέφερε αισθητές διαφορές στους χρόνους εκτέλεσης του προγράμματος σύγκρισης. Παρόλα αυτά για μεγάλα σύνολα δεδομένων το πρόγραμμα είναι ιδιαίτερα χρονοβόρο, όσον αφορά τουλάχιστον το υπολογιστικό σύστημα που χρησιμοποιήθηκε για τα πειράματα. Βέβαια οι χρόνοι

προετοιμασίας των δεδομένων, που περιλαμβάνει την σύσταση των μπλοκ, καθώς και οι χρόνοι κρυπτογράφησης, είναι παρόμοιοι με την περίπτωση μη εφαρμογής Blocking. Όλα τα προηγούμενα στοιχεία παρουσιάζονται στους πίνακες 5-7 ως 5-10.

| **Πλήθος Εγγραφών Συνόλων** | **Χρόνος σε sec δημιουργίας Τρι-γραμμάτων** | **Χρόνος σε sec κρυπτογρά-φησης** | **Χρόνος σε sec σύγκρισης** |
|---|---|---|---|
| 40×40 | 1 | 1 | 59 |
| 100×100 | 1 | 1 | 3,900 |
| 200×200 | 1 | 1 | 26,000 |
| 300×300 | 1 | 1 | 93,600 |
| 500×500 | 1 | 1 | 286,000 |
| 1,000×1,000 | 1 | 4 | ≈625 ώρες |
| 5,000×1,000 | 2 | 16 | ≈5,537.56 ώρες |
| 5,000×5,000 | 3 | 16 | ≈26,388.89 ώρες |

Πίνακας 5-7: Στατιστικά Στοιχεία Πρωτοκόλλου Χρήσης Τρι-γραμμάτων

| **Πλήθος Εγγραφών Συνόλων** | **Χρόνος σε sec δημιουργίας Τρι-γραμμάτων** | **Χρόνος σε sec κρυπτογρά-φησης** | **Χρόνος σε sec σύγκρισης** |
|---|---|---|---|
| 40×40 | 1 | 1 | 1 |
| 100×100 | 1 | 1 | 4 |
| 200×200 | 1 | 1 | 28 |
| 300×300 | 1 | 1 | 63 |
| 500×500 | 1 | 1 | 304 |
| 1,000×1,000 | 1 | 4 | 34.2 λεπτά |
| 5,000×1,000 | 2 | 16 | 275.32 λεπτά |
| 5,000×5,000 | 3 | 16 | 14.18 ώρες |

Πίνακας 5-8: Στατιστικά Στοιχεία Πρωτοκόλλου Χρήσης Τρι-γραμμάτων με Blocking

| Πλήθος Εγγραφών Συνόλων | Χρόνος σε sec δημιουργίας Τρι-γραμμάτων | Χρόνος σε sec κρυπτογρά-φησης | Χρόνος σε sec σύγκρισης |
|---|---|---|---|
| 40×40 | 1 | 1 | 7 |
| 100×100 | 1 | 1 | 89 |
| 200×200 | 1 | 1 | 600 |
| 300×300 | 1 | 1 | 2,100 |
| 500×500 | 1 | 1 | 8,500 |
| 1,000×1,000 | 1 | 1 | 16.67 ώρες |
| 5,000×1,000 | 1 | 1 | ≈338.89 ώρες |
| 5,000×5,000 | 1 | 1 | ≈1,611.12 ώρες |

Πίνακας 5-9: Στατιστικά Στοιχεία Τροποποιημένου Πρωτοκόλλου Χρήσης Τρι-γραμμάτων

| Πλήθος Εγγραφών Συνόλων | Χρόνος σε sec δημιουργίας Τρι-γραμμάτων | Χρόνος σε sec κρυπτογρά-φησης | Χρόνος σε sec σύγκρισης |
|---|---|---|---|
| 40×40 | 1 | 1 | 1 |
| 100×100 | 1 | 1 | 1 |
| 200×200 | 1 | 1 | 6 |
| 300×300 | 1 | 1 | 12 |
| 500×500 | 1 | 1 | 57 |
| 1,000×1,000 | 1 | 1 | 6.4 λεπτά |
| 5,000×1,000 | 1 | 1 | 35.25 λεπτά |
| 5,000×5,000 | 1 | 1 | 304.12 λεπτά |

Πίνακας 5-10: Στατιστικά Στοιχεία Τροποποιημένου Πρωτοκόλλου Χρήσης Τρι-γραμμάτων με Blocking

### 5.1.4 Πρωτόκολλο Χρήσης Απόστασης Levenshtein

Τα προγράμματα που υλοποιούν το πρωτόκολλο χρήσης της απόστασης Levenshtein είναι αρκετά γρηγορότερα από τις προηγούμενες περιπτώσεις. Η διάρκεια εκτέλεσης των προγραμμάτων προετοιμασίας και κρυπτογράφησης των δεδομένων είναι πολύ μικρή. Από την άλλη πλευρά, οι χρόνοι εκτέλεσης του προγράμματος σύγκρισης ποικίλλουν ανάλογα με το μέγεθος των συνόλων δεδομένων. Συγκεκριμένα μέχρι τις περιπτώσεις 1.000 εγγραφών οι χρόνοι είναι αρκετά ικανοποιητικοί, όμως στην περίπτωση των 5.000 εγγραφών, το πρόγραμμα καθυστερεί αρκετά. Παράλληλα, η εφαρμογή του Blocking επέφερε μια μικρή βελτίωση στην εκτέλεση του προγράμματος σύγκρισης. Όλα τα χρονικά αποτελέσματα διακρίνονται στους πίνακες 5-11 και 5-12.

| **Πλήθος Εγγραφών Συνόλων** | **Χρόνος σε sec προετοιμασίας** | **Χρόνος σε sec κρυπτογράφησης** | **Χρόνος σε sec σύγκρισης** |
|---|---|---|---|
| 100×100 | 1 | 1 | 37 |
| 200×200 | 1 | 1 | 145 |
| 300×300 | 1 | 1 | 343 |
| 500×500 | 1 | 1 | 933 |
| 1,000×1,000 | 1 | 4 | 1.08 ώρες |
| 5,000×1,000 | 2 | 16 | 5.27 ώρες |
| 5,000×5,000 | 3 | 16 | 26.38 ώρες |

Πίνακας 5-11: Στατιστικά Στοιχεία Πρωτοκόλλου Χρήσης Απόστασης Levenshtein

| Πλήθος Εγγραφών Συνόλων | Χρόνος σε sec προετοιμασίας | Χρόνος σε sec κρυπτογράφησης | Χρόνος σε sec σύγκρισης |
|---|---|---|---|
| 100×100 | 1 | 1 | 2 |
| 200×200 | 1 | 1 | 10 |
| 300×300 | 1 | 1 | 24 |
| 500×500 | 1 | 1 | 64 |
| 1,000×1,000 | 1 | 4 | 4.26 λεπτά |
| 5,000×1,000 | 2 | 16 | 20.61 λεπτά |
| 5,000×5,000 | 3 | 16 | 104.16 λεπτά |

Πίνακας 5-12: Στατιστικά Στοιχεία Πρωτοκόλλου Χρήσης Απόστασης Levenshtein με Blocking

### 5.1.5 Πρωτόκολλο Χρήσης Απόστασης Jaro - Winkler

Οι διαδικασίες του πρωτοκόλλου χρήσης της απόστασης Jaro – Winkler εκτελούνται πιο γρήγορα από όλα τα υπόλοιπα πρωτόκολλα. Τα προγράμματα προετοιμασίας και κρυπτογράφησης, όπως έχει αναφερθεί στις ενότητες 4.1.4 και 4.2.2, είναι τα ίδια με αυτά του πρωτοκόλλου που χρησιμοποιεί την απόσταση Levenshtein, οπότε έχουν τους ίδιους χρόνους εκτέλεσης. Σε αντίθεση όμως με τα προηγούμενα πρωτόκολλα, το πρόγραμμα σύγκρισης της ομοιότητας είναι αρκετά γρήγορο για όλα τα σύνολα δεδομένων που εξετάστηκαν. Ομοίως, ικανοποιητικά αποτελέσματα δίνει και η εφαρμογή του Blocking. Στο σημείο αυτό αξίζει να σημειωθεί ότι σε μια περίπτωση το πρόγραμμα με το Blocking έκανε περισσότερο χρόνο από το πρόγραμμα χωρίς. Επειδή όμως η διαφορά είναι μερικά δευτερόλεπτα, αυτό πρέπει να οφείλεται σε κάποια καθυστέρηση που προκάλεσε το υπολογιστικό σύστημα και όχι στην υλοποίηση του προγράμματος. Στους πίνακες 5-13 και 5-14 παρουσιάζονται όλα τα χρονικά αποτελέσματα για το πρωτόκολλο Jaro – Winkler.

| Πλήθος Εγγραφών Συνόλων | Χρόνος σε sec προετοιμασίας | Χρόνος σε sec κρυπτογράφησης | Χρόνος σε sec σύγκρισης |
|---|---|---|---|
| 100×100 | 1 | 1 | 9 |
| 200×200 | 1 | 1 | 36 |
| 300×300 | 1 | 1 | 82 |
| 500×500 | 1 | 1 | 227 |
| 1,000×1,000 | 1 | 4 | 15 λεπτά |
| 5,000×1,000 | 2 | 16 | 1.25 ώρες |
| 5,000×5,000 | 3 | 16 | 6.25 ώρες |

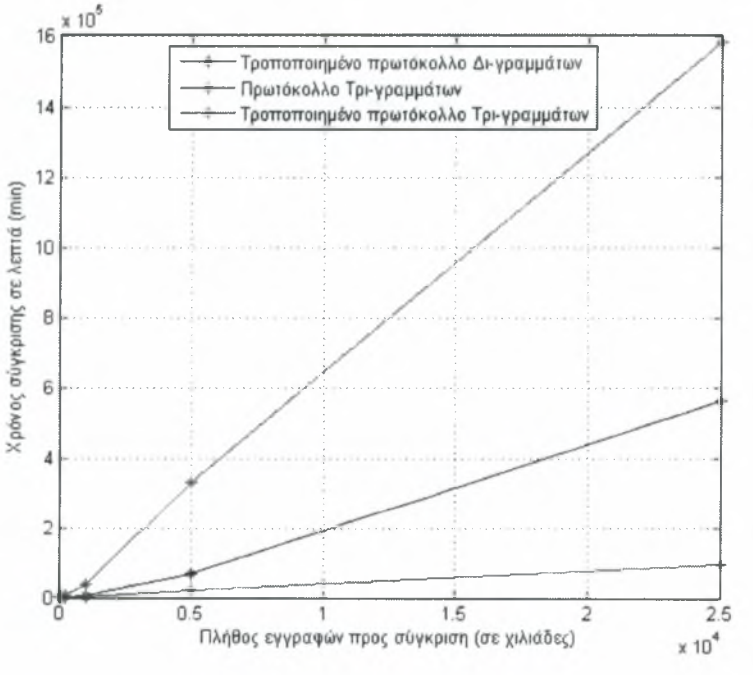
Πίνακας 5-13: Στατιστικά Στοιχεία Πρωτοκόλλου Χρήσης Απόστασης Jaro – Winkler

| Πλήθος Εγγραφών Συνόλων | Χρόνος σε sec προετοιμασίας | Χρόνος σε sec κρυπτογράφησης | Χρόνος σε sec σύγκρισης |
|---|---|---|---|
| 100×100 | 1 | 1 | 3 |
| 200×200 | 1 | 1 | 10 |
| 300×300 | 1 | 1 | 21 |
| 500×500 | 1 | 1 | 58 |
| 1,000×1,000 | 1 | 4 | 3.93 λεπτά |
| 5,000×1,000 | 2 | 16 | 18.9 λεπτά |
| 5,000×5,000 | 3 | 16 | 83.34 λεπτά |

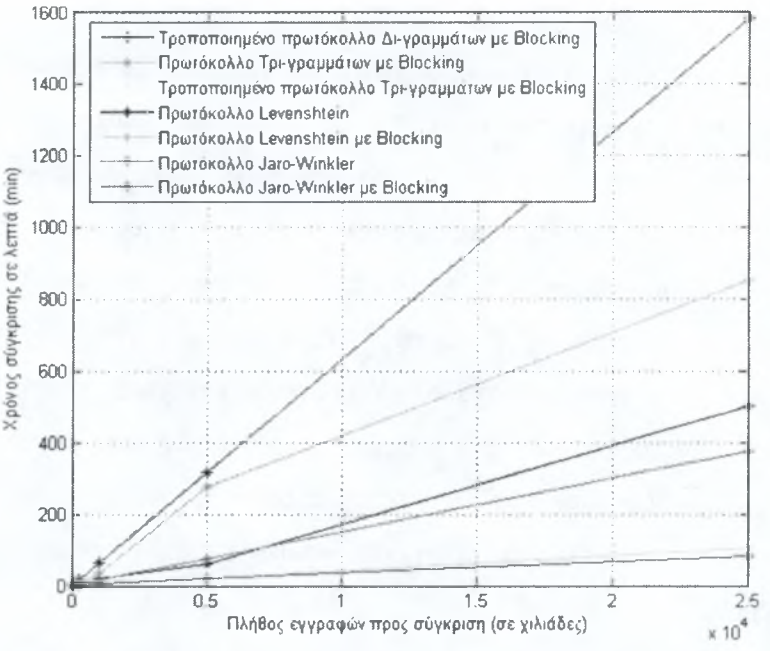
Πίνακας 5-14: Στατιστικά Στοιχεία Πρωτοκόλλου Χρήσης Απόστασης Jaro – Winkler με Blocking

### 5.1.6 Γραφικές Παραστάσεις Πρωτοκόλλων

Σε αυτήν την ενότητα παρουσιάζονται οι γραφικές παραστάσεις των χρονικών αποτελεσμάτων των πρωτοκόλλων σε δύο διαφορετικά γραφήματα. Συγκεκριμένα στο σχήμα 5-1 παρουσιάζονται οι γραφικές παραστάσεις του τροποποιημένου πρωτοκόλλου για Δι-γράμματα, και των δύο πρωτοκόλλων για Τρι-γράμματα. Στο σχήμα 5-2 παρουσιάζονται οι γραφικές παραστάσεις των τριών προαναφερθέντων πρωτοκόλλων με χρήση Blocking, καθώς και οι γραφικές παραστάσεις των πρωτοκόλλων Levenshtein και Jaro - Winkler με, και χωρίς, χρήση Blocking. Ο λόγος που δεν τοποθετήθηκαν όλες οι γραφικές παραστάσεις στο ίδιο γράφημα είναι ότι οι δύο πρώτες εμφανίζουν πολύ υψηλές τιμές στον άξονα του χρόνου, σε σχέση με τις υπόλοιπες, με αποτέλεσμα ορισμένες γραφικές παραστάσεις να ήταν ιδιαίτερα δυσδιάκριτες.




Σχήμα 5-1: Γράφημα χρονοβόρων πρωτοκόλλων




Σχήμα 5-2: Γράφημα λιγότερο χρονοβόρων πρωτοκόλλων

## 5.2 Αξιολόγηση Αποτελεσμάτων Ταιριάσματος

Σε αυτήν την ενότητα θα εξεταστούν τα πειραματικά αποτελέσματα από την σκοπιά του ταιριάσματος των δεδομένων. Δηλαδή θα παρουσιαστούν στοιχεία τα οποία υποδεικνύουν κατά πόσο τα πρωτόκολλα που περιγράφθηκαν στο κεφάλαιο 4 μπορούν να επιτύχουν ορθά την διασύνδεση γνωρισμάτων. Στην κατεύθυνση αυτή εξετάστηκαν συγκεκριμένα σύνολα δεδομένων τα οποία περιελάμβαναν εγγραφές με προκαθορισμένες διαφορές στα αλφαριθμητικά των πεδίων τους. Οι διαφορές αυτές θεωρήθηκε ότι είναι οι κατάλληλες, ώστε κάποιος να κρίνει την απόδοση του εκάστοτε πρωτοκόλλου.

Προκειμένου να γίνει σωστά η αξιολόγηση χρησιμοποιήθηκαν και μεγάλα και μικρά αλφαριθμητικά. Αυτά προέρχονται από τα πεδία «επώνυμο» και «όνομα» των συνόλων δεδομένων. Οι διαφοροποιήσεις που έγιναν μεταξύ παρόμοιων αλφαριθμητικών είναι οι εξής:

- Προσθήκη ενός χαρακτήρα σε κάποιο σημείο του αλφαριθμητικού.
- Αφαίρεση ενός χαρακτήρα από οποιοδήποτε σημείο του αλφαριθμητικού.
- Αλλαγή ενός χαρακτήρα του αλφαριθμητικού με κάποιον άλλον.
- Αλλαγή ενός χαρακτήρα με κάποιον άλλον, και αφαίρεση ενός άλλου χαρακτήρα από το αλφαριθμητικό.

### 5.2.1 Πρωτόκολλο Χρήσης Ν-γραμμάτων

Για το πρωτόκολλο χρήσης Ν-γραμμάτων εξετάστηκαν οι περιπτώσεις για *N = 2, 3, 4, 5*. Όπως φαίνεται στους πίνακες 5-15 ως 5-18, τα αποτελέσματα που δίνει η σύγκριση σε όλες τις περιπτώσεις έχουν αρκετά μικρή ποσοστιαία βαθμολογία. Μάλιστα το μεγαλύτερο ποσοστό που επιτυγχάνεται είναι περίπου *0,66* για την σύγκριση δύο αλφαριθμητικών όπου το ένα έχει έναν παραπάνω χαρακτήρα στο τέλος. Οι μικρές βαθμολογίες δείχνουν ότι το πρωτόκολλο θα αντιμετωπίζει μια δυσκολία στο ταίριασμα γνωρισμάτων που έχουν λίγες αλλά συχνά εμφανιζόμενες διαφοροποιήσεις. Για παράδειγμα αλφαριθμητικά με δύο διαφορετικούς χαρακτήρες ενδέχεται να έχουν ποσοστό ταιριάσματος *0.16* περίπου, που είναι ιδιαίτερα χαμηλό.

Επίσης, παρατηρώντας το πλήθος των Ν-γραμμάτων που παράγονται, καθώς και τα κοινά Ν-γράμματα των αλφαριθμητικών, συνάγεται ένα ενδιαφέρον συμπέρασμα. Όταν διαφέρουν κατά έναν χαρακτήρα στην άκρη των αλφαριθμητικών, τότε περίπου τα μισά Ν-γράμματα θα είναι διαφορετικά. Αν η διαφοροποίηση εμφανίζεται σε χαρακτήρα στο μέσον περίπου των αλφαριθμητικών, τότε θα διαφέρουν τα *3/4* των Ν-γραμμάτων. Προφανώς από αυτό αποδεικνύεται η ιδιαίτερα μικρή ποσοστιαία βαθμολογία που παρουσιάζουν τα ζεύγη αλφαριθμητικών. Παράλληλα ένα τελευταίο στοιχείο που παρατηρείται είναι ότι κάθε φορά που αυξάνεται το μήκος των Ν-γραμμάτων κατά ένα, τότε το πλήθος των παραγόμενων Ν-γραμμάτων υποδιπλασιάζεται. Αυτό είναι λογικό αν αναλογιστεί κανείς τον τύπο "*πλήθος* = $2^a - 1$" που αναφέρεται στην ενότητα 3.2.

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Κοινά Ν-γράμματα | Ν-γράμματα 1ης Εγγραφής | Ν-γράμματα 2ης Εγγραφής | Ποσοστιαία Βαθμολογία |
|---|---|---|---|---|---|
| smith | smith | 15 | 15 | 15 | 1 |
| smith | smithe | 15 | 15 | 31 | 0.6521 |
| johnson | johnsen | 15 | 63 | 63 | 0.2380 |
| williams | wiliames | 31 | 127 | 127 | 0.2440 |
| miller | moler | 3 | 31 | 15 | 0.1304 |
| cunningham | caningham | 63 | 511 | 255 | 0.1644 |
| montgomery | montegomery | 255 | 511 | 1023 | 0.3324 |
| stevenson | stevensone | 255 | 255 | 511 | 0.6657 |
| alan | alain | 3 | 7 | 15 | 0.2727 |
| jennifer | jenifer | 63 | 127 | 63 | 0.6631 |
| robert | robberto | 31 | 31 | 127 | 0.3924 |
| margaret | margeret | 63 | 127 | 127 | 0.4960 |
| richard | rickarde | 15 | 63 | 127 | 0.1578 |
| charles | charkes | 15 | 63 | 63 | 0.2380 |
| linda | lindy | 7 | 15 | 15 | 0.4666 |

Πίνακας 5-15: Συγκριτικά αποτελέσματα για Δι-γράμματα

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Κοινά Ν-γράμματα | Ν-γράμματα 1ης Εγγραφής | Ν-γράμματα 2ης Εγγραφής | Ποσοστιαία Βαθμολογία |
|---|---|---|---|---|---|
| smith | smith | 7 | 7 | 7 | 1 |
| smith | smithe | 7 | 7 | 15 | 0.6363 |
| johnson | johnsen | 7 | 31 | 31 | 0.2258 |
| williams | wiliames | 7 | 63 | 63 | 0.1111 |
| miller | moler | 1 | 15 | 7 | 0.0909 |
| cunningham | caningham | 31 | 255 | 127 | 0.1623 |
| montgomery | montegomery | 63 | 255 | 511 | 0.1644 |
| stevenson | stevensone | 127 | 127 | 255 | 0.664 |
| alan | alain | 1 | 3 | 7 | 0.2 |
| jennifer | jenifer | 15 | 63 | 31 | 0.3191 |
| robert | robberto | 7 | 15 | 63 | 0.1794 |
| margaret | margeret | 7 | 63 | 63 | 0.1111 |
| richard | rickarde | 3 | 31 | 63 | 0.0638 |
| charles | charkes | 3 | 31 | 31 | 0.0967 |
| linda | lindy | 3 | 7 | 7 | 0.4285 |

Πίνακας 5-16: Συγκριτικά αποτελέσματα για Τρι-γράμματα

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Κοινά Ν-γράμματα | Ν-γράμματα 1ης Εγγραφής | Ν-γράμματα 2ης Εγγραφής | Ποσοστιαία Βαθμολογία |
|---|---|---|---|---|---|
| smith | smith | 3 | 3 | 3 | 1 |
| smith | smithe | 3 | 3 | 7 | 0.6 |
| johnson | johnsen | 3 | 15 | 15 | 0.2 |
| williams | wiliames | 1 | 31 | 31 | 0.0322 |
| miller | moler | 0 | 7 | 3 | 0 |
| cunningham | caningham | 15 | 127 | 63 | 0.1578 |
| montgomery | montegomery | 15 | 127 | 255 | 0.0785 |
| stevenson | stevensone | 63 | 63 | 127 | 0.6631 |
| alan | alain | 0 | 1 | 3 | 0 |
| jennifer | jenifer | 3 | 31 | 15 | 0.1304 |
| robert | robberto | 1 | 7 | 31 | 0.0526 |
| margaret | margeret | 1 | 31 | 31 | 0.0322 |
| richard | rickarde | 0 | 15 | 31 | 0 |
| charles | charkes | 1 | 15 | 15 | 0.0666 |
| linda | lindy | 1 | 3 | 3 | 0.3333 |

Πίνακας 5-17: Συγκριτικά αποτελέσματα για Τετρα-γράμματα

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Κοινά N-γράμματα | N-γράμματα 1ης Εγγραφής | N-γράμματα 2ης Εγγραφής | Ποσοστιαία Βαθμολογία |
|---|---|---|---|---|---|
| smith | smith | 1 | 1 | 1 | 1 |
| smith | smithe | 1 | 1 | 3 | 0.5 |
| johnson | johnsen | 1 | 7 | 7 | 0.1428 |
| williams | wiliames | 0 | 15 | 15 | 0 |
| miller | moler | 0 | 3 | 1 | 0 |
| cunningham | caningham | 7 | 63 | 31 | 0.1489 |
| montgomery | montegomery | 3 | 63 | 127 | 0.0315 |
| stevenson | stevensone | 31 | 31 | 63 | 0.6595 |
| alan | alain | 0 | 1 | 1 | 0 |
| jennifer | jenifer | 1 | 15 | 7 | 0.0909 |
| robert | robberto | 0 | 3 | 15 | 0 |
| margaret | margeret | 0 | 15 | 15 | 0 |
| richard | rickarde | 0 | 7 | 15 | 0 |
| charles | charkes | 0 | 7 | 7 | 0 |
| linda | lindy | 0 | 1 | 1 | 0 |

Πίνακας 5-18: Συγκριτικά αποτελέσματα για Πεντα-γράμματα

### 5.2.2 Τροποποιημένο Πρωτόκολλο Χρήσης Ν-γραμμάτων

Για το τροποποιημένο πρωτόκολλο εξετάστηκαν οι ίδιες κατηγορίες Ν-γραμμάτων με το αρχικό πρωτόκολλο. Το συμπέρασμα που εξάγεται είναι ότι εκτός από ταχύτερο, το τροποποιημένο πρωτόκολλο Ν-γραμμάτων δίνει και καλύτερα ποσοστά ταιριάσματος. Πλέον στις συγκρίσεις που γίνονται, η μεγαλύτερη ποσοστιαία βαθμολογία είναι της τάξης του *0,88* για σχετικά μεγάλα αλφαριθμητικά για Δι-γράμματα. Μάλιστα η βαθμολογία αυτή παραμένει σε υψηλά επίπεδα, της τάξης του *0,83*, μέχρι και στην περίπτωση των Πεντα-γραμμάτων. Στην περίπτωση των μικρών αλφαριθμητικών η μέγιστη ποσοστιαία βαθμολογία φτάνει το *0,8* για Δι-γράμματα, αλλά πέφτει στο *0,5* για Πεντα-γράμματα. Προφανώς η βελτίωση των αποτελεσμάτων σύγκρισης οφείλεται στο γεγονός ότι δεν υπολογίζεται το δυναμοσύνολο των Ν-γραμμάτων, αλλά μόνο οι υπο-λίστες με μήκος ένα και δύο.

Εν τέλει από τα αποτελέσματα παρατηρεί κανείς ότι το πρωτόκολλο λειτουργεί καλύτερα για Δι-γράμματα και Τρι-γράμματα παρά για τις υπόλοιπες περιπτώσεις. Επίσης είναι δυνατόν να οριστεί ένα κατώφλι της τάξης του *0,4* για εγγραφές που θεωρείται ότι ταιριάζουν και *0,2* για εγγραφές που θεωρείται ότι πιθανόν ταιριάζουν. Όλα τα προηγούμενα φαίνονται στους πίνακες 5-19 ως 5-22.

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Κοινά Ν-γράμματα | Ν-γράμματα 1ης Εγγραφής | Ν-γράμματα 2ης Εγγραφής | Ποσοστιαία Βαθμολογία |
|---|---|---|---|---|---|
| smith | smith | 10 | 10 | 10 | 1 |
| smith | smithe | 10 | 10 | 15 | 0.8 |
| johnson | johnsen | 10 | 21 | 21 | 0.4761 |
| williams | wiliames | 15 | 28 | 28 | 0.5357 |
| miller | moler | 3 | 15 | 10 | 0.24 |
| cunningham | caningham | 21 | 45 | 36 | 0.5185 |
| montgomery | montegomery | 36 | 45 | 55 | 0.72 |
| stevenson | stevensone | 36 | 36 | 45 | 0.8889 |
| alan | alain | 3 | 6 | 10 | 0.375 |
| jennifer | jenifer | 21 | 28 | 21 | 0.8571 |
| robert | robberto | 15 | 15 | 28 | 0.6976 |
| margaret | margeret | 15 | 21 | 28 | 0.6122 |
| richard | rickarde | 10 | 21 | 28 | 0.4081 |
| charles | charkes | 10 | 21 | 21 | 0.4761 |
| linda | lindy | 6 | 10 | 10 | 0.6 |

Πίνακας 5-19: Συγκριτικά αποτελέσματα Τροποποιημένου Πρωτοκόλλου για Δι-γράμματα

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Κοινά Ν-γράμματα | Ν-γράμματα 1ης Εγγραφής | Ν-γράμματα 2ης Εγγραφής | Ποσοστιαία Βαθμολογία |
|---|---|---|---|---|---|
| smith | smith | 6 | 6 | 6 | 1 |
| smith | smithe | 6 | 6 | 10 | 0.75 |
| johnson | johnsen | 6 | 15 | 15 | 0.4 |
| williams | wiliames | 6 | 21 | 21 | 0.2857 |
| miller | moler | 1 | 10 | 6 | 0.125 |
| cunningham | caningham | 15 | 36 | 28 | 0.4687 |
| montgomery | montegomery | 21 | 36 | 45 | 0.5185 |
| stevenson | stevensone | 28 | 28 | 36 | 0.875 |
| alan | alain | 1 | 3 | 6 | 0.2222 |
| jennifer | jenifer | 10 | 21 | 15 | 0.5556 |
| robert | robberto | 6 | 10 | 21 | 0.387 |
| margaret | margeret | 6 | 21 | 21 | 0.2857 |
| richard | rickarde | 3 | 15 | 21 | 0.1667 |
| charles | charkes | 3 | 15 | 15 | 0.2 |
| linda | lindy | 3 | 6 | 6 | 0.5 |

Πίνακας 5-20: Συγκριτικά αποτελέσματα Τροποποιημένου Πρωτοκόλλου για Τρι-γράμματα

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Κοινά Ν-γράμματα | Ν-γράμματα 1ης Εγγραφής | Ν-γράμματα 2ης Εγγραφής | Ποσοστιαία Βαθμολογία |
|---|---|---|---|---|---|
| smith | smith | 3 | 3 | 3 | 1 |
| smith | smithe | 3 | 3 | 6 | 0.6667 |
| johnson | johnsen | 3 | 10 | 10 | 0.3 |
| williams | wiliames | 1 | 15 | 15 | 0.0667 |
| miller | moler | 0 | 6 | 3 | 0 |
| cunningham | caningham | 10 | 28 | 21 | 0.4081 |
| montgomery | montegomery | 10 | 28 | 36 | 0.3125 |
| stevenson | stevensone | 21 | 21 | 28 | 0.8571 |
| alan | alain | 0 | 1 | 3 | 0 |
| jennifer | jenifer | 3 | 15 | 10 | 0.24 |
| robert | robberto | 1 | 6 | 15 | 0.095 |
| margaret | margeret | 1 | 15 | 15 | 0.067 |
| richard | rickarde | 0 | 10 | 15 | 0 |
| charles | charkes | 1 | 10 | 10 | 0.1 |
| linda | lindy | 1 | 3 | 3 | 0.3333 |

Πίνακας 5-21: Συγκριτικά αποτελέσματα Τροποποιημένου Πρωτοκόλλου για Τετρα-γράμματα

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Κοινά Ν-γράμματα | Ν-γράμματα 1ης Εγγραφής | Ν-γράμματα 2ης Εγγραφής | Ποσοστιαία Βαθμολογία |
|---|---|---|---|---|---|
| smith | smith | 1 | 1 | 1 | 1 |
| smith | smithe | 1 | 1 | 3 | 0.5 |
| johnson | johnsen | 1 | 6 | 6 | 0.1667 |
| williams | wiliames | 0 | 10 | 10 | 0 |
| miller | moler | 0 | 3 | 1 | 0 |
| cunningham | caningham | 6 | 21 | 15 | 0.3333 |
| montgomery | montegomery | 3 | 21 | 28 | 0.1224 |
| stevenson | stevensone | 15 | 15 | 21 | 0.8333 |
| alan | alain | 0 | 1 | 1 | 0 |
| jennifer | jenifer | 1 | 10 | 6 | 0.125 |
| robert | robberto | 0 | 3 | 10 | 0 |
| margaret | margeret | 0 | 10 | 10 | 0 |
| richard | rickarde | 0 | 6 | 10 | 0 |
| charles | charkes | 0 | 6 | 6 | 0 |
| linda | lindy | 0 | 1 | 1 | 0 |

Πίνακας 5-22: Συγκριτικά αποτελέσματα Τροποποιημένου Πρωτοκόλλου για Πεντα-γράμματα

### 5.2.3 Πρωτόκολλο Χρήσης Τρι-γραμμάτων

Το πρωτόκολλο χρήσης Τρι-γραμμάτων παρουσιάζει πολύ καλά αποτελέσματα όσον αφορά το ταίριασμα των γνωρισμάτων. Σύμφωνα με όσα ειπώθηκαν στην ενότητα 3.3, κάθε ζεύγος γνωρισμάτων των εγγραφών θα έχει διαφορετικό κατώφλι. Όπως φαίνεται

στον πίνακα 5-23, για κάθε ζεύγος αλφαριθμητικών με μικρές διαφοροποιήσεις, η ένταση της διαφοράς τους είναι μικρότερη από το εκάστοτε κατώφλι, επομένως θεωρούνται ότι ταιριάζουν. Μόνο σε μια περίπτωση δύο μικρών αλφαριθμητικών που έχουν μικρή διαφορά αλλά αντιστοιχούν σε διαφορετικές οντότητες, το πρωτόκολλο θεώρησε λανθασμένα ότι ταιριάζουν. Επίσης στον πίνακα 5-23 διακρίνεται ότι όμοια αλφαριθμητικά έχουν μηδενική ένταση διαφοράς, ενώ αλφαριθμητικά με δύο ή τρεις διαφορετικούς χαρακτήρες έχουν ένταση διαφοράς κοντά στην τιμή κατωφλίου.

| **Αλφαριθμητικό 1ης Εγγραφής** | **Αλφαριθμητικό 2ης Εγγραφής** | **Ένταση Διαφοράς** | **Κατώφλι** |
|---|---|---|---|
| smith | smith | 0 | 2.736 |
| smith | smithe | 1 | 2.586 |
| johnson | johnsen | 2 | 2.661 |
| williams | wiliames | 2.4494 | 2.711 |
| miller | moler | 2.2360 | 2.636 |
| cunningham | caningham | 2.2360 | 2.736 |
| montgomery | montegomery | 2.2360 | 2.761 |
| stevenson | stevensone | 1 | 2.686 |
| alan | alain | 1.7320 | 2.586 |
| jennifer | jenifer | 1.4142 | 2.661 |
| robert | robberto | 2 | 2.661 |
| margaret | margeret | 2.4494 | 2.711 |
| richard | rickarde | 2.6457 | 2.711 |
| charles | charkes | 2.4494 | 2.686 |
| linda | lindy | 1.4142 | 2.586 |
| alan | linda | 2.6457 | 2.661 |

Πίνακας 5-23: Συγκριτικά αποτελέσματα για το Δεύτερο Πρωτόκολλο

### 5.2.4 Πρωτόκολλο Χρήσης Απόστασης Levenshtein

Ιδιαίτερη επιτυχία στο ταίριασμα των γνωρισμάτων παρουσιάζει το πρωτόκολλο που χρησιμοποιεί την απόσταση Levenshtein. Μέσα από τον πίνακα 5-24 μπορεί κάποιος να διαπιστώσει ότι το πρωτόκολλο επιτυγχάνει ορθό ταίριασμα γνωρισμάτων που έχουν μικρές διαφοροποιήσεις αλλά αντιστοιχούν στην ίδια οντότητα. Στον ίδιο πίνακα διακρίνεται εκτός από το πλήθος των αντιμεταθέσεων, και η τιμή κατωφλίου που έχει κάθε ζεύγος γνωρισμάτων των εγγραφών.

Όπως έχει προαναφερθεί στην ενότητα 3.4, ο τρόπος υπολογισμού του κατωφλίου δεν υπήρχε στον αλγόριθμο αλλά ορίστηκε κατά την διάρκεια της έρευνας. Αυτό συνέβη διότι μετρώντας μόνο το πλήθος των αντιμεταθέσεων υπήρχε περίπτωση να ταιριάξουν μικρά αλφαριθμητικά που δεν αντιστοιχούν στην ίδια οντότητα. Πλέον με τη χρήση του κατωφλίου λαμβάνεται υπόψιν, εκτός από το πλήθος των αντιμεταθέσεων, και το μήκος των αλφαριθμητικών που συγκρίνονται. Επομένως ορίζοντας ένα κατώφλι $\theta \leq 1/5$ το πρωτόκολλο μπορεί να ταιριάξει σωστά όλα τα μεγέθη αλφαριθμητικών.

Όπως είναι φανερό δύο όμοια αλφαριθμητικά θα έχουν μηδενικό αριθμό αντιμεταθέσεων και τιμή κατωφλίου $\theta = 0$. Διαφορετικά αλφαριθμητικά τα οποία όμως ταιριάζουν έχουν από μια ως δύο αντιμεταθέσεις και η τιμή κατωφλίου τους κυμαίνεται από τιμές κοντά στο μηδέν ως τιμές που προσεγγίζουν το $0.2$. Παράλληλα, όταν συγκρίνονται μεγάλα αλφαριθμητικά, είναι πολύ πιθανό να θεωρηθεί ότι ταιριάζουν ακόμη και αν το πλήθος των μεταξύ τους αντιμεταθέσεων είναι $3$ ή $4$. Μάλιστα παρουσιάζει ενδιαφέρον το γεγονός ότι ένα ζεύγος μικρών αλφαριθμητικών με δύο αντιμεταθέσεις, έχει την ίδια τιμή κατωφλίου με ένα ζεύγος μεγάλων αλφαριθμητικών με τέσσερις αντιμεταθέσεις.

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Πλήθος Αντιμεταθέσεων | Κατώφλι |
|---|---|---|---|
| smith | smith | 0 | 0 |
| smith | smithe | 1 | 0.0909 |
| johnson | johnsen | 1 | 0.0714 |
| williams | wiliames | 2 | 0.125 |
| miller | moler | 2 | 0.1818 |
| cunningham | caningham | 2 | 0.1052 |
| montgomery | montegomery | 1 | 0.0476 |
| montgomery | montegomeree | 3 | 0.1363 |
| mentgomery | montegomeree | 4 | 0.1818 |
| stevenson | stevensone | 1 | 0.0526 |
| alan | alain | 1 | 0.1111 |
| jennifer | jenifer | 1 | 0.0666 |
| robert | robberto | 2 | 0.1428 |
| margaret | margeret | 1 | 0.0625 |
| richard | rickarde | 2 | 0.1333 |
| charles | charkes | 1 | 0.0714 |
| linda | lindy | 1 | 0.1 |

Πίνακας 5-24: Συγκριτικά αποτελέσματα για το Τρίτο Πρωτόκολλο

### 5.2.5 Πρωτόκολλο Χρήσης Απόστασης Jaro - Winkler

Ομοίως με το προηγούμενο πρωτόκολλο, αυτό που χρησιμοποιεί την απόσταση Jaro – Winkler επιτυγχάνει ορθώς την διασύνδεση των γνωρισμάτων. Το συγκεκριμένο πρωτόκολλο παρουσιάζει ιδιαίτερη επιτυχία στο ταίριασμα σχετικά μικρών αλφαριθμητικών, χωρίς αυτό να σημαίνει ότι δεν αποδίδει καλά και για μεγαλύτερα αλφαριθμητικά.

Όπως διαφαίνεται στον πίνακα 5-25, δύο όμοια αλφαριθμητικά θα έχουν τιμή κατωφλίου ίση με ένα, ενώ δύο τελείως διαφορετικά θα έχουν μηδενική τιμή κατωφλίου. Ανεξαρτήτως του μήκους των προς σύγκριση αλφαριθμητικών, ένα ζεύγος θεωρείται ότι ταιριάζει αν ξεπερνά την προκαθορισμένη τιμή κατωφλίου $\theta \geq 0.8$, όπως έχει προαναφερθεί στην ενότητα 3.5.

Παράλληλα στον πίνακα γίνεται εμφανές ότι τα μικρότερα ζεύγη αλφαριθμητικών πετυχαίνουν υψηλότερες τιμές από ότι τα μεγαλύτερα. Εξαίρεση αποτελούν τα ζεύγη με μεγάλο μήκος που έχουν το μεγαλύτερο πρόθεμα. Γενικά ισχύει ότι όσο μεγαλύτερο είναι το κοινό πρόθεμα ενός ζεύγους αλφαριθμητικών, τόσο υψηλότερες είναι οι τιμές που επιτυγχάνει το πρωτόκολλο. Το γεγονός αυτό επιβεβαιώνεται αν συγκριθούν οι τιμές κατωφλίου που έχουν τα ζεύγη "*montgomery - montegomery*" και "*stevenson - stevensone*", όπου και τα δύο διαφέρουν κατά έναν χαρακτήρα και έχουν το ίδιο μήκος. Όμως στο πρώτο το κοινό πρόθεμα είναι *3* ενώ στο δεύτερο είναι *9*, με αποτέλεσμα το δεύτερο ζεύγος να παρουσιάζει υψηλότερη τιμή κατωφλίου.

| Αλφαριθμητικό 1ης Εγγραφής | Αλφαριθμητικό 2ης Εγγραφής | Κατώφλι |
|---|---|---|
| smith | smith | 1 |
| smith | smithe | 0.9666 |
| johnson | johnsen | 0.9428 |
| williams | wiliames | 0.9416 |
| miller | moler | 0.84 |
| cunningham | caningham | 0.8691 |
| montgomery | montegomery | 0.9418 |
| stevenson | stevensone | 0.98 |
| alan | alain | 0.9533 |
| jennifer | jenifer | 0.9708 |
| robert | robberto | 0.9416 |
| margaret | margeret | 0.95 |
| richard | rickarde | 0.9083 |
| charles | charkes | 0.9428 |
| linda | lindy | 0.92 |

Πίνακας 5-25: Συγκριτικά αποτελέσματα για το Τέταρτο Πρωτόκολλο

## 5.3 Συμπεράσματα

Μετά την εκτέλεση όλων των πειραμάτων δημιουργήθηκε μια γενική εικόνα για την συνολική απόδοση όλων των πρωτοκόλλων που υλοποιήθηκαν κατά την διάρκεια της έρευνας. Πλέον ο χρήστης μπορεί να εντοπίσει τα πλεονεκτήματα και τα μειονεκτήματα του κάθε πρωτοκόλλου, και είναι σε θέση να κρίνει ποιο πρωτόκολλο

θα εφαρμόσει στα δεδομένα του ώστε να έχει τα καλύτερα δυνατά αποτελέσματα. Το πιο σημαντικό στοιχείο που διακρίνεται μέσα από τα πειράματα είναι ότι όλα τα πρωτόκολλα παράγουν ακριβώς τα ίδια αποτελέσματα διασύνδεσης γνωρισμάτων, είτε εφαρμόζονται σε κρυπτογραφημένα είτε σε μη κρυπτογραφημένα δεδομένα.

Ένα πρώτο συμπέρασμα που συνάγεται από τα πειράματα στις ενότητες 5.1 και 5.2 είναι ότι το πρωτόκολλο που εφαρμόζει τον αλγόριθμο του Jaro – Winkler είναι το πιο αποδοτικό σε γενικές γραμμές. Το συγκεκριμένο πρωτόκολλο, είτε εφαρμόζει Blocking των εγγραφών είτε όχι, επιτυγχάνει τους ταχύτερους χρόνους εκτέλεσης σε σχέση με τα υπόλοιπα. Παράλληλα είναι σε θέση να συνδέσει ορθώς τις εγγραφές που αντιστοιχούν στην ίδια οντότητα του πραγματικού κόσμου, και δεν υποπίπτει σε λανθασμένα ταιριάσματα.

Το πρωτόκολλο χρήσης Ν-γραμμάτων παρουσιάζει αρκετά μειονεκτήματα σε όλους τους τομείς. Το πιο βασικό μειονέκτημα αφορά την ικανότητα του στη διασύνδεση των γνωρισμάτων. Οι ποσοστιαίες βαθμολογίες είναι πολύ μικρές με αποτέλεσμα να μην μπορεί να επιβεβαιωθεί η ομοιότητα ή όχι των δεδομένων. Βέβαια αν αναλογιστεί κανείς ότι στην περίπτωση εγγραφών που αναφέρονται σε διαφορετικές οντότητες η ποσοστιαία βαθμολογία είναι σχεδόν μηδενική, είναι δυνατόν, με την παρέμβαση του χρήστη, να ελεγχθεί κατά πόσο ένα σχετικά μικρό ποσοστό ομοιότητας ενδέχεται να αφορά εγγραφές που αναφέρονται στην ίδια οντότητα. Παράλληλα, η όλη διαδικασία είναι ιδιαίτερα χρονοβόρα λόγω της δημιουργίας των δυναμοσυνόλων. Επιπλέον κατά την επεξεργασία των δεδομένων παράγεται ιδιαίτερα μεγάλος όγκος Ν-γραμμάτων. Αρκεί μόνο να σκεφτεί κανείς ότι *4* εγγραφές παράγουν πάνω από *20.000* διαφορετικά Ν-γράμματα, ενώ *40* εγγραφές παράγουν πάνω από *1.000.000* διαφορετικά Ν-γράμματα.

Σε αντίθεση με το προηγούμενο, το τροποποιημένο πρωτόκολλο εμφανίζει αισθητές βελτιώσεις. Τα αποτελέσματα της διασύνδεσης είναι αρκετά ικανοποιητικά, ιδιαίτερα στην περίπτωση των Δι-γραμμάτων και Τρι-γραμμάτων, οπότε στην πλειονότητα των περιπτώσεων είναι δυνατή η επιβεβαίωση της ομοιότητας των εγγραφών. Παράλληλα αυτή η έκδοση του πρωτοκόλλου έχει την δυνατότητα να

αντιμετωπίσει σύνολα δεδομένων οποιουδήποτε μεγέθους, παρότι η διαδικασία της σύγκρισης είναι αρκετά χρονοβόρα.

Το πρωτόκολλο χρήσης Τρι-γραμμάτων έχει σχεδόν παρόμοια απόδοση με το τροποποιημένο πρωτόκολλο Ν-γραμμάτων. Όπως φαίνεται στην ενότητα 5.2.3, το συγκεκριμένο πρωτόκολλο έχει αρκετά καλά ποσοστά ταιριάσματος με εξαίρεση την περίπτωση των σχετικά μικρών αλφαριθμητικών, όπου υπάρχει η πιθανότητα να ταιριάξει λανθασμένα αλφαριθμητικά με λίγες διαφοροποιήσεις που αντιστοιχούν σε διαφορετικές οντότητες. Αντίστοιχα, όσον αφορά τους χρόνους εκτέλεσης του, η εφαρμογή του πρωτοκόλλου σε μεγάλα σύνολα δεδομένων είναι απαγορευτική όταν εξετάζονται και τα *13* πεδία κάθε εγγραφής. Στην περίπτωση όμως της εξέτασης μόνο των δύο πεδίων υπάρχει μια αισθητή βελτίωση που καθιστά το πρωτόκολλο περισσότερο λειτουργικό.

Το πρωτόκολλο που υλοποιεί τον αλγόριθμο του Levenshtein είναι αρκετά αποδοτικό. Ιδιαίτερα στην επιτυχία ταιριάσματος των δεδομένων, μέσα από αυτήν την έρευνα αποδείχθηκε ότι η συγκεκριμένη μέθοδος μπορεί να δώσει το ίδιο καλά αποτελέσματα για τα κρυπτογραφημένα και μη δεδομένα. Επιπλέον η όλη διαδικασία εκτέλεσης των προγραμμάτων παρουσιάζει μια σχετική καθυστέρηση, χωρίς όμως να καθιστά απαγορευτική την εφαρμογή του πρωτοκόλλου σε μεγάλα σύνολα δεδομένων.

Στο σημείο αυτό αξίζει να τονισθεί η διαφορά στους χρόνους εκτέλεσης των πειραμάτων όταν οι εγγραφές οργανώνονται σε μπλοκ. Σε όλα τα πρωτόκολλα που εξετάστηκαν υπήρξε ραγδαία μείωση των χρόνων εκτέλεσης τους, καθιστώντας τα πιο λειτουργικά για μεγάλα σύνολα δεδομένων. Μόνο στο πρωτόκολλο χρήσης Ν-γραμμάτων με δυναμοσύνολα δεν υπήρχε η δυνατότητα εξέτασης του με Blocking, αφού μπορούσε να αντιμετωπίσει μόνο σύνολα δεδομένων με *40* εγγραφές. Βέβαια το αδύναμο σημείο αυτής της διαδικασίας είναι ότι υπάρχει περίπτωση δύο εγγραφές που αντιστοιχούν στην ίδια οντότητα και δεν ανήκουν στο ίδιο μπλοκ, να μην εξεταστούν ποτέ, κάτι το οποίο λύνεται με τις επαναληπτικές εφαρμογές του Blocking με διαφορετικά κλειδιά. Επομένως ο χρήστης οφείλει να βρει την χρυσή τομή, ανάλογα με τις απαιτήσεις του, ανάμεσα στην αύξηση του χρόνου εκτέλεσης και στα μεγαλύτερα ποσοστά επιτυχίας ταιριάσματος.

Τέλος, όπως προαναφέρθηκε στην αρχή του κεφαλαίου, τα πειράματα εκτελέστηκαν σε ένα απλό υπολογιστικό σύστημα. Είναι εμφανές λοιπόν ότι αν εκτελεστούν σε μεγάλα δίκτυα υπολογιστών ή σε υπερ-υπολογιστές, που συνήθως διαθέτουν οι οργανισμοί που θα προβούν στην σύγκριση μεγάλων συνόλων δεδομένων, οι χρόνοι εκτέλεσης θα είναι κατά πολύ μικρότεροι. Αυτό σημαίνει ότι και τα μη λειτουργικά πρωτόκολλα, θα μπορούν να εφαρμοστούν σε μεγάλα σύνολα δεδομένων.

# Κεφάλαιο 6

# Επίλογος

Η ασφαλής διασύνδεση εγγραφών αποτελεί έναν σχετικά σύγχρονο και συνεχώς αναπτυσσόμενο τομέα των Βάσεων Δεδομένων. Συνεχώς παρουσιάζονται νέες έρευνες στις οποίες προσπαθούν, εφαρμόζοντας διάφορες μεθόδους, να λύσουν το πρόβλημα της κατανάλωσης πολλών υπολογιστικών πόρων, αλλά και να επιτύχουν την διατήρηση της εμπιστευτικότητας των δεδομένων. Πολλές από τις εφαρμογές που υλοποιούνται είναι επιτυχημένες ενώ άλλες παρουσιάζουν ορισμένα προβλήματα.

Στην παρούσα εργασία εξετάστηκε ένα συγκεκριμένο πλαίσιο ασφάλειας στο οποίο εφαρμόστηκαν διαφορετικές μέθοδοι διασύνδεσης γνωρισμάτων, που οδηγούν στην διασύνδεση εγγραφών. Όπως διαφαίνεται μέσα από τα πειράματα που εκτελέστηκαν κατά την διάρκεια της έρευνας, το πρωτόκολλο που χρησιμοποιεί N-γράμματα με δυναμοσύνολα αντιμετωπίζει ιδιαίτερα προβλήματα τόσο στην ικανότητα να ταιριάζει ορθά τα γνωρίσματα που αντιστοιχούν στην ίδια οντότητα, όσο και στην χρονική διάρκεια επεξεργασίας των δεδομένων. Μάλιστα δεν πρέπει να παραβλεφθεί το γεγονός ότι δεν υπήρχε η δυνατότητα να επεξεργαστεί μεγάλα σύνολα δεδομένων. Για τον λόγο αυτό υλοποιήθηκε η τροποποιημένη έκδοση του πρωτοκόλλου, καθώς και το πρωτόκολλο χρήσης Τρι-γραμμάτων, τα οποία μπορούν να εφαρμοστούν σε οποιοδήποτε μέγεθος συνόλων δεδομένων με αρκετή επιτυχία στο ταίριασμα των γνωρισμάτων. Το μόνο πρόβλημα αυτών των μεθόδων εντοπίζεται στην διάρκεια εκτέλεσης των προγραμμάτων σύγκρισης της ομοιότητας. Από την άλλη πλευρά την καλύτερη συμπεριφορά παρουσιάζουν τα δύο πρωτόκολλα που κάνουν χρήση των μετρικών απόστασης Levenshtein και Jaro – Winkler. Αυτά έχουν ιδιαίτερη ικανότητα

στο να ταιριάζουν σωστά τα γνωρίσματα και οι χρόνοι εκτέλεσης τους κυμαίνονται σε λογικά πλαίσια.

Παράλληλα, προκειμένου να λυθεί το πρόβλημα του χρόνου εφαρμόστηκε σε όλα τα παραπάνω πρωτόκολλα μια διαδικασία Blocking με την οποία η σύγκριση γίνεται μόνο μεταξύ εγγραφών που ανήκουν στο ίδιο μπλοκ. Η συγκεκριμένη διαδικασία είχε ως αποτέλεσμα να μειωθούν σε μεγάλο ποσοστό οι χρόνοι όλων των πρωτοκόλλων. Αυτό βέβαια έχει ως τίμημα την πιθανότητα να μην ταιριάξουν εγγραφές που αντιστοιχούν στην ίδια οντότητα αν δεν ανήκουν στο ίδιο μπλοκ.

Ιδιαίτερο ενδιαφέρον παρουσιάζουν ορισμένες μελλοντικές επεκτάσεις της έρευνας. Εκτός από τις μεθόδους που υλοποιήθηκαν θα ήταν ενδιαφέρον να εξεταστεί η απόδοση του πρωτοκόλλου των Ravikumar et al [34], όπου γίνεται χρήση μετρικών απόστασης όπως το TFIDF, το SoftTFIDF και η Ευκλείδεια απόσταση. Εξίσου ενδιαφέρουσα είναι οι πρόταση των C. O' Keefe et al [29] οι οποίοι προτείνουν ορισμένα πρωτόκολλα διασύνδεσης και εξαγωγής δεδομένων με χρήση κρυπτογραφίας και συναρτήσεων κατακερματισμού. Ένα χαρακτηριστικό στοιχείο των πρωτοκόλλων τους είναι ότι, κάθε συμβαλλόμενο μέρος μπορεί να παράγει κοινά ψευδώνυμα των δεδομένων του, αποτρέποντας την αποκάλυψη οποιασδήποτε ευαίσθητης πληροφορίας στα υπόλοιπα μέρη του πρωτοκόλλου. Επίσης θα μπορούσε να γίνει μια σύγκριση των πρωτοκόλλων που επιτυγχάνουν την ασφαλή διασύνδεση εγγραφών χρησιμοποιώντας κρυπτογραφικές μεθόδους με αυτά που δεν κάνουν χρήση κρυπτογραφίας. Ένα παράδειγμα τέτοιου πρωτοκόλλου έχει υλοποιηθεί από τους Scannapieco et al [36] όπου χρησιμοποιούν την μέθοδο SparseMap ώστε να εισαγάγουν τις εγγραφές προς σύγκριση στο Ευκλείδειο διάστημα.

Γενικότερα είναι δυνατή η εφαρμογή ενός πλαισίου διατήρησης της ασφάλειας, όπως αυτό που προτάθηκε στην εργασία, σε οποιοδήποτε εργαλείο διασύνδεσης εγγραφών, όπως είναι το Febrl [9] και το TAILOR [15]. Εκτός των άλλων, θα μπορούσε να δημιουργηθεί ένα εργαλείο ασφαλούς διασύνδεσης εγγραφών το οποίο να αποτελείται από τα προγράμματα που υλοποιήθηκαν κατά την διάρκεια της εργασίας.

Συνοψίζοντας, το συμπέρασμα που εξάγεται από την εργασία είναι ότι συνεχώς αναπτύσσονται ενδιαφέρουσες προτάσεις στον τομέα της ασφαλούς διασύνδεσης

εγγραφών. Για τον λόγο αυτό πρέπει όλες οι προτάσεις να εξετάζονται προσεκτικά και μεθοδικά ώστε να κρίνεται η ικανότητα τους ως προς την επίτευξη του στόχου τους.

# Βιβλιογραφία


[1] A. Al-Lawati, D. Lee and P. McDaniel, "Blocking-Aware Private Record Linkage", *in Proceedings of IQIS '05, Baltimore*, MD, USA, 2005.

[2] M. Atallah, H. Elmongui, V. Deshpande, and L. Schwarz, "Secure supply-chain protocols", in *IEEE International Conference on E-Commerce,* Newport Beach, California, June 24-27 2003, pp. 293-302.

[3] R.Baxter, P. Christen and T. Churches, "A Comparison of Fast Blocking Methods for Record Linkage", in *Proceedings of the first Workshop on Data Cleaning, Record Linkage and Object Consolidation, KDD 2003, Washington DC*, August 24-27 2003.

[4] M. Bilenko, B. Kamath, R. J. Mooney, "Adaptive Blocking: Learning to Scale Up Record Linkage" in *Proceedings of The sixth IEEE International Conference on Data Mining (ICDM),* Hong Kong, 2006, pp. 87-96.

[5] N. Bruno, L. Gravano, and A. Marian, "Evaluating Top-k Queries over Web-Accessible Databases", in ACM Transactions on Database Systems TODS archive, volume 29, issue 2, June 2004, pp. 319-362.

[6] P. Christen, "Probabilistic Data Generation for Deduplication and Data Linkage", in *Proceedings of The sixth IDEAL 2005 International Conference,* Brisbane, Australia, July 6-8 2005, pp. 109-116.

[7] P. Christen, "Privacy-Preserving Data Linkage and Geocoding: Current Approaches and Research Directions", in *Proceedings of the Workshop on Privacy Aspects of Data Mining ( PADM),* Hong Kong, December 2006.

[8] P. Christen, "Towards Parameter-free Blocking for Scalable Record Linkage", ANU Joint Computer Science Technical Report Series, TR-CS-07-03, August 2007.

[9] P. Christen, and T. Churches, "Febrl: Release 0.3", Documentation, Australian National University, April 6 2005.

[10] T. Churches, and P. Christen, "Blind Data Linkage using n-gram Similarity Comparisons" in *Proceedings of the eighth PAKDD 2004 Conference,* Sydney, Australia, May 26-28 2004, pp. 121-126.

[11] C. Clifton, A. Doan, A. Elmagarmid, M. Kantarcioglu, G. Schadow, D. Suciu, and J. Vaidya, "Privacy-Preserving Data Integration and Sharing", in *Proceedings of DMKD 2004 Conference.* Paris, France, June 13 2004, pp. 19-26.

[12] W. Cohen, "The WHIRL Approach to Integration: An Overview", in *Proceedings of the AAAI-98 Workshop on AI and Information Integration*, AAAI Press, 1998

[13] W. Cohen, P. Ravikumar, and S. E. Fienberg, "A Comparison of String Distance Metrics for Name-Matching Tasks", in *Proceedings of IJCAI 2003 Workshop on Information Integration on the Web,* August 9-10 2003, Acapulco, Mexico, pp. 73-78.

[14] I. Dinur, and K. Nissim, "Revealing Information while Preserving Privacy", in *Proceedings of PODS 2003 Conference.* San Diego, California, USA, June 9-12 2003, pp. 202-210.

[15] M. Elfeky, V. Verykios, and A. Elmagarmid, "TAILOR: A record linkage toolbox", in *Proceedings of the 18th International Conference on Data Engineering ICDE 2002,* San Jose, California, Feb. 2002.

[16] Federal Information Processing Standards Publication 197 (FIPS 197) "Announcing the ADVANCED ENCRYPTION STANDARD (AES)", November 26, 2001.

[17] I. Fellegi, and A. Sunter, "A Theory for Record Linkage", in Journal of the American Statistical Society, 1969.

[18] B. Gedik, and L. Liu, "A Customizable k-Anonymity Model for Protecting Location Privacy", in Georgia Institute of Technology – CERCS-04-15 Technical Reports.

[19] K. Goiser, P. Christen, "Towards Automated Record Linkage" in *Proceedings of the fifth Australasian Conference on Data Mining and Analytics, Sydney,* Australia, 2006, pp. 23-31.

[20] O. Goldreich, "The Foundations of Cryptography – Volume 2", Cambridge University Press, 2004.

[21] L. Gravano, P. G. Ipeirotis, H.V. Jagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, and D. Srivastava, "Using q-grams in a DBMS for Approximate String Processing", in IEEE Data Engineering Bulletin 24(4), 2001, pp 145-154.

[22] M. Hernandez and S. Stolfo, "The Merge/Perge problem for Large Databases", in *Proceedings of the SIGMOD Conference,* San Jose, 1995

[23] J. A. Hylton, "Identifying and Merging Related Bibliographic Records", in MIT Technical Report TR-678, June 1996.

[24] I. F. Ilyas, W.G. Aref, and A. K. Elmagarmid, "Supporting top-k join queries in relational databases", in *Proceedings of the 29th International Conference on Very Large Databases, VLDB 2003,* Berlin, Germany, pp. 754-765.

[25] M. Jaro, "Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida", Journal of the American Statistical Society, 84(406):414-420, 1989.

[26] M. Kantarcioglu, J. Jin, and C. Clifton, "When do Data Mining Results Violate Privacy?", in *Proceedings of KDD 2004 Conference,* Seattle, Washington, USA, August 22-25 2004, pp. 599-604.

[27] A.J Lait, and B. Randell, "An Assessment of Name Matching Algorithms", Technical Report, Department of Computing S cience, University of Newcastle upon Tyne, UK 1993.

[28] M. Michelson, C. A. Knoblock, "Learning Blocking Schemes for Record Linkage", American Association for Artificial Intelligence, 2006.

[29] C. M. O' Keefe, M. Yung, L. Gu, and R. Baxter, "Privacy-Preserving Data Linkage Protocols", in *Proceedings of 2004 ACM Workshop on Privacy in the electronic society,* Washington DC, USA, 2004, pp. 94-102.

[30] G. Pagalos, and I. Maurides, "Security in Informational Systems and Networks", 2002, pp. 187-215.

[31] C. Pang, D. Hansen, "Improved Record Linkage for Encrypted Identifying Data", E-Health Research Centre, ICT Centre, CSIRO, Brisbane, Australia.

[32] Paull, "D.L.: A geocoded National Address File for Australia: The G-NAF What, Why, Who and When?", PSMA Australia Limited, Griffith, ACT, Australia, 2003.

[33] L. Philips, "The Double-Metaphone Search Algorithm", C/C++ User's Journal, vol. 18 no. 6, June 2000.

[34] P. Ravikumar, W. Cohen, , and S. E. Fienberg, "A Secure Protocol for Computing String Distance Metrics", in *Proceedings of Workshop on Privacy and Security Aspects of Data Mining ICDM 2004.*

[35] D. Sankoff and J. Kruskal, "Time warps, string edits, and macromolecules: the theory and practice of sequence comparison", editors, CSLI Publications, 1999, pp 1-44.

[36] M. Scannapieco, I. Figotin, E. Bertino and A. Elmagarmid, "Schema and Data Privacy Preserving Record Matching", in *Proceedings of the 32nd VLDB Conference, Seoul*, Korea, 2006.

[37] B. Schneier, J. kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, "Twofish: A 128-Bit Block Cipher", June 15, 1998

[38] D. Struck, "Don't store my data, Japanese tell government", International Herald Tribune, p.1, August 24-25 2002.

[39] F. Tsui, J. Espino, V. Dato, P. Gesteland, J. Hutman, and M. Wagner, "Technical description of RODS: A real-time public health surveillance system", J AM Inform Association, vol. 10, no. 5, pp. 399-408, September 2003.

[40] J. Vaidya, and C. Clifton, "Privacy-Preserving Association Rule Mining in Vertically Partitioned Data", in *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 23-26 2002, pp. 639-644.

[41] J. Vaidya, and C. Clifton, "Privacy-Preserving k-means Clustering over Vertically Partitioned Data", in *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* Washington, DC, August 24-27 2003, pp. 206-213.

[42] J. Vaidya, and C. Clifton, "Privacy-Preserving Naïve Bayes Classifier for Vertically Partitioned Data".

[43] V. Verykios, A. Elmagarmid, M. Elfeky, M. Cochinwala and S. Dalal, "On the Completeness and Accuracy of the Record Matching Process", in *Proceedings of the MIT Conference on Information Quality*, Boston, MA, October 2000.

[44] V. Verykios, A. Elmagarmid, and E. Houstis, "Automating the Approximate Record-Matching Process", Information Sciences, vol. 126, July 2000.

[45] V. Verykios, G. Moustakides, and M. Elfeky, "A Bayesian decision model for cost optimal record matching", *The Very Large Data Bases Journal,* vol. 12, no. 1, pp. 28-40, May 2003.

[46] W. Winkler, "The State of Record Linkage and Current Research Problems", Statistics of Income Division, Internal Revenue Service Publication R99/04.

[47] W. Winkler and Y. Thibaudeau, "An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 U.S. Decennial Census", U.S. Bureau of the Census, Statistical Research Division Technical Report RR91/09.

[48] L. Xiong, S. Chitti, and L. Liu, "Topk Queries across Multiple Private Databases", in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems ICDCS 2005,* pp. 145-154.

[49] S. Yan, D. Lee, M. Y. Kan, and C. L. Giles, "Adaptive Sorted Neighborhood Methods for Efficient Record Linkage", in *Proceedings of the 2007 Conference on Digital Libraries JCDL,* Vancouver, British Columbia, Canada, June 2007, pp. 185-194.

[50] S. Zhong, Z. Yang, and R. N. Wright, "Privacy-Enhancing k-Anonymization of Customer Data", in *Proceedings of the 24th ACM SIGMOD 2005 symposium on Principles of database systems,* Baltimore, Maryland, USA, pp. 139-147.

[51] Β. Μητρογιάννης, «Διατήρηση της Εμπιστευτικότητας κατά την Ενοποίηση των Δεδομένων σε Κατανεμημένες Βάσεις Δεδομένων», Διπλωματική Εργασία, Πανεπιστήμιο Θεσσαλίας, Βόλος, Σεπτέμβριος 2005.

[52] The Febrl Project Web site. http://datamining.anu.edu.au/projects/linkage.html.

[53] The TwoFish Cryptographic Algorithm. http://www.twofish.org.

[54] The MD5 Message-Digest Algorithm. http://www.faqs.org/rfcs/rfc1321.html.

[55] The Python programming language Web site. http://www.python.org.

[56] The Perl programming language Web site. http://www.perl.org.

[57] The Perl Programs Library and Documentation Web site. http://www.cpan.org.

[58] The AES cryptosystem Web site. http://www.cryptosystem.net/aes/

[59] The RSA Security System. http://www.rsasecurity.com.

[60] The U.S. Bureau of the Census Web site. http://www.census.gov/

[61] The G-NAF Web site. http://www.g-naf.com.au/

# Παράρτημα

Στο συνοδευτικό DVD περιέχονται όλα τα προγράμματα και το λογισμικό που χρησιμοποιήθηκαν κατά την έρευνα.

Συγκεκριμένα στον φάκελο \Programs συμπεριλαμβάνονται όλα τα perl αρχεία που υλοποιήθηκαν για τις εφαρμογές των πρωτοκόλλων όπως περιγράφθηκαν στο κεφάλαιο 4. Αυτά είναι διαχωρισμένα στους υπο-φακέλους \Without_Block και \With_Block, οι οποίοι με τη σειρά τους χωρίζονται στους υπο-φακέλους \Ngrams, \Ngrams_Mod, \Trigrams, \Edit_Distance και \Jaro_Winkler. Προκειμένου κάποιος να εφαρμόσει ένα πρωτόκολλο πρέπει να εκτελέσει τα αντίστοιχα προγράμματα με συγκεκριμένη σειρά δίνοντας στον kernel τις παρακάτω εντολές.

- Για το πρωτόκολλο χρήσης Ν-γραμμάτων και την τροποποιημένη έκδοση:
  1. perl create_ngrams.plx inputA.csv outputA.csv N ή
     perl create_ngrams_mod.plx inputA.csv outputA.csv N

Δέχεται σαν παραμέτρους το πρώτο σύνολο δεδομένων, το αρχείο με τα Ν-γράμματα, και το μήκος των Ν-γραμμάτων.

  2. perl create_ngrams.plx inputB.csv outputB.csv N ή
     perl create_ngrams_mod.plx inputB.csv outputB.csv N

Ομοίως για το δεύτερο σύνολο δεδομένων. Το Ν πρέπει να είναι ίδιο και για τις δύο περιπτώσεις.

  3. perl encr_ngrams.plx outputA.csv 16 a123456789123456
     perl encr_ngrams_mod.plx outputA.csv 16 a123456789123456

Δέχεται σαν παραμέτρους το αρχείο με τα Ν-γράμματα, το μήκος του κλειδιού και το κλειδί κρυπτογράφησης.

  4. perl encr_ngrams.plx outputB.csv 16 a123456789123456

perl encr_ngrams_mod.plx outputB.csv 16 a123456789123456

Ομοίως για το αρχείο με τα Ν-γράμματα του δεύτερου συνόλου.

5. perl sort_encrypted.plx croutputA.csv
6. perl sort_encrypted.plx croutputB.csv

Δέχεται σαν παράμετρο τα κρυπτογραφημένα αρχεία.

7. perl comp_ngrams.plx scroutputA.csv scroutputB.csv skeyinputA.csv skeyinputB.csv results.txt F

Δέχεται σαν παραμέτρους τα δύο ταξινομημένα κρυπτογραφημένα αρχεία με τα Ν-γράμματα, τα δυο ταξινομημένα αρχεία με τα κλειδιά των εγγραφών, το αρχείο όπου θα αποθηκευτούν τα αποτελέσματα και τον αριθμό των πεδίων που έχει κάθε εγγραφή, που είναι *13* για το αρχικό πρωτόκολλο και *2* για το τροποποιημένο.

- Για το πρωτόκολλο χρήσης Τρι-γραμμάτων:

1. perl create_trigrams.plx inputA.csv outputA.csv

Δέχεται σαν παραμέτρους το πρώτο σύνολο δεδομένων και το αρχείο με τα Τρι-γράμματα που παράγεται.

2. perl create_trigrams.plx inputB.csv outputB.csv

Ομοίως για το δεύτερο σύνολο δεδομένων.

3. perl encr_trigrams.plx outputA.csv 16 a123456789123456

Δέχεται σαν παραμέτρους το αρχείο με τα Τρι-γράμματα, το μήκος του κλειδιού και το κλειδί κρυπτογράφησης.

4. perl encr_trigrams.plx outputB.csv 16 a123456789123456

Ομοίως για το αρχείο με τα Τρι-γράμματα του δεύτερου συνόλου.

5. perl sort_encrypted.plx croutputA.csv
6. perl sort_encrypted.plx croutputB.csv

Δέχεται σαν παράμετρο τα κρυπτογραφημένα αρχεία.

7. perl comp_trigrams.plx scroutputA.csv scroutputB.csv skeyinputA.csv skeyinputB.csv results.txt F

Δέχεται σαν παραμέτρους τα δύο ταξινομημένα κρυπτογραφημένα αρχεία με τα Τρι-γράμματα, τα δυο ταξινομημένα αρχεία με τα κλειδιά των εγγραφών, το αρχείο όπου

θα αποθηκευτούν τα αποτελέσματα και τον αριθμό των πεδίων που έχει κάθε εγγραφή, που είναι *13* για το αρχικό πρωτόκολλο και *2* για το τροποποιημένο.

- Για το πρωτόκολλο χρήσης της απόστασης Levenshtein:

1. perl create_edit_dist.plx inputA.csv outputA.csv

Δέχεται σαν παραμέτρους το πρώτο σύνολο δεδομένων και το αρχείο που παράγεται.

2. perl create_ edit_dist.plx inputB.csv outputB.csv

Ομοίως για το δεύτερο σύνολο δεδομένων.

3. perl encr_ edit_dist.plx outputA.csv 16 a123456789123456

Δέχεται σαν παραμέτρους το αρχείο που έχει παραχθεί, το μήκος του κλειδιού και το κλειδί κρυπτογράφησης.

4. perl encr_ edit_dist.plx outputB.csv 16 a123456789123456

Ομοίως για το παραγόμενο αρχείο του δεύτερου συνόλου.

5. perl comp_edit_dist.plx scroutputA.csv scroutputB.csv skeyinputA.csv skeyinputB.csv results.txt F

Δέχεται σαν παραμέτρους τα δύο ταξινομημένα κρυπτογραφημένα αρχεία, τα δυο ταξινομημένα αρχεία με τα κλειδιά των εγγραφών, το αρχείο όπου θα αποθηκευτούν τα αποτελέσματα και τον αριθμό των πεδίων που έχει κάθε εγγραφή.

- Για το πρωτόκολλο χρήσης της απόστασης Jaro - Winkler:

1. perl create_jaro.plx inputA.csv outputA.csv

Δέχεται σαν παραμέτρους το πρώτο σύνολο δεδομένων και το αρχείο που παράγεται.

2. perl create_ jaro.plx inputB.csv outputB.csv

Ομοίως για το δεύτερο σύνολο δεδομένων.

3. perl encr_ jaro.plx outputA.csv 16 a123456789123456

Δέχεται σαν παραμέτρους το αρχείο που έχει παραχθεί, το μήκος του κλειδιού και το κλειδί κρυπτογράφησης.

4. perl encr_ jaro.plx outputB.csv 16 a123456789123456

Ομοίως για το παραγόμενο αρχείο του δεύτερου συνόλου.

5. perl comp_jaro.plx scroutputA.csv scroutputB.csv skeyinputA.csv skeyinputB.csv results.txt F

Δέχεται σαν παραμέτρους τα δύο ταξινομημένα κρυπτογραφημένα αρχεία, τα δυο ταξινομημένα αρχεία με τα κλειδιά των εγγραφών, το αρχείο όπου θα αποθηκευτούν τα αποτελέσματα και τον αριθμό των πεδίων που έχει κάθε εγγραφή.

Είναι προφανές ότι αν ο χρήστης θελήσει να εφαρμόσει Blocking θα εκτελέσει τις παραπάνω εντολές για τα αντίστοιχα προγράμματα του υπο-φακέλου \With_Block.

Στον φάκελο \Software περιλαμβάνονται τα αρχεία για την εγκατάσταση των γλωσσών προγραμματισμού Perl και Python καθώς και το λογισμικό Cygwin. Η γλώσσα Python χρειάζεται για να λειτουργήσει το Febrl. Κανονικά τα προγράμματα εκτελούνται σε περιβάλλον Linux/Unix. Παράλληλα όμως είναι δυνατόν να εκτελεστούν και σε περιβάλλον Windows με την χρήση του λογισμικού Cygwin. Οι εντολές για την χρήση του είναι ίδιες με αυτές που χρησιμοποιούνται στο Unix. Η εγκατάσταση των παραπάνω είναι απλή, αλλά σε περίπτωση δυσκολίας υπάρχουν τα αρχεία readme.txt που διαθέτουν οδηγίες και διάφορες πληροφορίες.

Στον φάκελο \Febrl υπάρχει ένα συμπιεσμένο αρχείο που περιλαμβάνει το λογισμικό Febrl. Αυτό για να λειτουργήσει πρέπει να αποσυμπιεστεί μέσα στον φάκελο που είναι εγκατεστημένη η γλώσσα προγραμματισμού Python. Στο φάκελο υπάρχει και το αρχείο febrldoc-0.3.pdf που αποτελεί τον οδηγό για την χρησιμοποίηση των διαφόρων εφαρμογών του λογισμικού.

Στον φάκελο \Crypto υπάρχουν τα modules του αλγορίθμου κρυπτογράφησης και της συνάρτησης κατακερματισμού. Αυτά για να χρησιμοποιηθούν πρέπει να εγκατασταθούν μέσα στο Cygwin.