National Technical University of Athens
School of Electrical and Computer Engineering

Master's Thesis

Big Data Anonymization

Author:
Nikolaos Dimakopoulos

Supervisor:
Prof. Andreas-Georgios Stafylopatis

Acknowledgments

In concluding my dissertation at the Laboratory of Artificial Intelligence and Learning Systems (AILS) for the postgraduate program of Data Science and Machine Learning at the National Technical University of Athens (NTUA), I would like to thank I would like to warmly thank those who helped me to complete this master's thesis.

First, I would like to thank prof. Andreas-Georgios Stafylopatis, for whom I am grateful for my thesis's supervision and the help that he provided to me.

Next, I would like to thank Dr. Manolis Terrovitis, Senior Researcher at the Information Systems Management Institute of Research Center Athena, who helped me and taught me data privacy fundamentals. He was also a key contributor to the development of the algorithm described in this current thesis, offering me solutions and ideas to various problems, and I am grateful to him for his full support and cooperation.

In addition, I owe another thank you to the Ph.D. candidate Dimitris Tsitsigos for the appropriate resources he provided me, while his contribution was significant to the algorithm's development.

Moreover, I owe my gratitude to Dr. Danae Pla-Karidi for her assistance in writing and elaborating the present study. The excellent cooperation we had both in the context of my thesis and our joint presence at the Research Center Athena helped me to face and overcome various difficulties that arose.

Furthermore, I would like to thank my fellow students for their invaluable help during an unprecedented period due to the pandemic, which made our study even more difficult.

Nevertheless, I owe the biggest thanks to my parents, Giannis and Panagiota, who sacrificed all these years to support my undergraduate studies at the National and Kapodistrian University of Athens and my postgraduate studies at NTUA and for the help they have offered me all these years.

Finally, I would like to dedicate this master's thesis to my fiancée Vicky, my sister Maria, and my aunt and second mother, Dimitra, for their tireless support, mainly on a psychological level.

Nikolaos Dimakopoulos
Athens, July 2022

Extended Abstract

The present era is the era of information and direct communication in any part of the world. Government documents, transactions, and our communication are all digital, and the huge volume of data is stored very easily and cheaply and is available to anyone for processing. The significant development of the internet and technology has led us to a digital age; now, the world is moving into a digital reality, which poses many risks to protecting our privacy. How can we protect our privacy? The EU, since May 2018, has established a mechanism for the protection of personal data in each member country, the Data Protection Privacy Regulation, also known as the GDPR, to avoid unconsented sharing of personal information. Nevertheless, is a strict mechanism sufficient to protect sensitive information? Clearly not, because any malicious combination of free data from various sources can target an individual, while hackers can violate the security systems of companies and institutions and steal huge volumes of personal data.

In addition, the growth of technology and, more specifically, the progress of artificial intelligence and machine learning is inextricably linked to big data and its processing. However, now researchers and scientists must also consider the data protection from various malicious people who try to process it for their purposes. So, what is the solution to protect privacy?

The answer is data anonymization. Anonymization of personal data is the process of encrypting or removing personally identifiable data from data sets so that the person can no longer be identified directly or indirectly. So, data anonymization is not limited to simply removing the identification of an individual, such as name or surname, but it is focused on ”hiding” secondary information such as age, zip code, gender, etc. The data may include pieces of information that are not themselves unique identifiers but can become identifying when combined with other datasets. These pieces are known as quasi-identifiers. For example, around 87 percent of the US population can be uniquely identified with just their 5-digit zip code, gender, and date of birth taken together. Even in cases where only a small fraction of individuals are uniquely identifiable, it can still lead to a severe privacy breach for the individuals affected. It is never possible to know the full set of what additional information is out there and, therefore, what could be identifying.

Data anonymization reduces the risk of unintended disclosure when sharing data between countries, industries, and even departments within the same company. It also reduces the chances of identity theft occurring. Anonymization of data is performed in various ways, including deletion, encryption, generalization, and others. A company can either delete personally identifiable information (PII) from its gathered data or encrypt this information with a strong passphrase. They can also decide to generalize the information collected in their database.

Various data protection models have been developed, such as k-anonymity, l-diversity, D-volatility, differential privacy, etc., which anonymize the data while limiting the information loss. The anonymization procedure must take into account the utility of the anonymized data. It is important for a dataset not to lose its utility after the anonymization completely

in order for the statisticians and analysts to be able to draw safe conclusions about the data. For example, a table contains the exact gross income earned by five CEOs in the retail sector. Let's assume the recorded incomes are $520,000, $230,000, $109,000, $875,000, and $124,000. This information can be generalized into categories like "< $500,000" and "≥ $500,000". The generalization categories are given through a hierarchy, a structured grouping of entities that share common attributes. It is a powerful and widely used method for representing common characteristics among entities while preserving their differences. Although the data is obfuscated, it will still be useful to the user. Decoding anonymized data is possible through a process known as De-anonymization (or "re-identification"). Since anonymized data can be decoded and unraveled, critics believe anonymization provides a false sense of security. For instance, different attack methods re-identify a k-anonymous table because k-anonymity does not provide adequate protection against attribute exposure. However, k—anonymity is a common and most popular model that preserves privacy. For K-anonymity to be achieved in a dataset, there need to be at least k-records that share the set of attributes that might reveal the entity's identity whose properties are represented in the dataset. Nevertheless, l-diversity came to solve the problems that arose from k—anonymity. The main purpose is that there should be diversity in sensitive data values greater than or equal to l for each subgroup created by k-anonymity. However, l— diversity also has limitations and issues that t-closeness helps resolve. Many other anonymization techniques will be analyzed in this thesis, and they try to figure out other problems; however, they are outdated.

On the other hand, there is a relatively new and promising technique that sharply reduces information loss and preserves intelligence privacy. Differential privacy is a mathematical definition of what it means to have privacy and addresses the paradox of learning nothing about an individual while learning useful information about a population. Nonetheless, there are limitations to the anonymization of big data. The huge volume and complexity characterize big data, and it is very demanding to handle them in the main memory of a traditional operating system. Modern computing systems provide the speed, power, and flexibility needed to access massive amounts and types of big data quickly. Along with reliable access, companies also need methods for integrating the data, building data pipelines, ensuring data quality, providing data governance and storage, and preparing the data for analysis. Traditional data tools are not equipped to handle this kind of complexity and volume, which has led to a slew of specialized big data software and architecture solutions designed to manage the load. The diversity of big data makes it inherently complex, resulting in the need for systems capable of processing its various structural and semantic differences. So, most handling and processing operations take place on the hard disk, where specialized databases can store the data in a way that does not require strict adherence to a particular model. Big data is used in nearly every industry to identify patterns and trends, answer questions, gain insights into customers, and tackle complex problems. Companies and organizations use the information for many reasons like growing their businesses, understanding customer decisions, enhancing research, making forecasts, and targeting key audiences for advertising, so big data anonymization must prevent privacy leakages.

In the current thesis, we focused on big data anonymization, implementing an algorithm that holds the main volume of data into the hard disk and in the main memory handles basic information about them. Following a one-time clustering approach in order to be executed

in regular terms of time, the records are deposited into clusters using a similarity-distance metric. After that, the clusters which contain similar-identical are anonymized through the hierarchies and k-anonymity privacy model. This is a demanding algorithm, and except for execution time and information loss, we have to handle clusters' splitting, "small" (size less than k) clusters, fast iteration through thousands of clusters, etc. The disk-based clustering algorithm is developed in the context of the Amnesia Anonymization Tool, which provides many anonymization techniques via a user-friendly graphical user interface. Finally, the algorithm's evaluation took place in Amnesia Anonymization Tool using the information loss metrics *NCP* and *Total*.

Εκτεταμένη Ελληνική Περίληψη

Η σημερινή εποχή είναι η εποχή της πληροφορίας και της άμεσης επικοινωνίας σε οποιοδήποτε μέρος του κόσμου. Τα κρατικά έγγραφα, οι συναλλαγές και η επικοινωνία μας είναι όλα ψηφιακά και ο τεράστιος όγκος δεδομένων αποθηκεύεται πολύ εύκολα και φθηνά και είναι διαθέσιμος σε οποιονδήποτε για επεξεργασία. Η σημαντική ανάπτυξη του διαδικτύου και της τεχνολογίας μας έχει οδηγήσει σε μια ψηφιακή εποχή. Τώρα, ο κόσμος κινείται σε μια ψηφιακή πραγματικότητα, η οποία εγκυμονεί πολλούς κινδύνους για την προστασία του απορρήτου μας. Πώς μπορούμε να προστατεύσουμε το απόρρητό μας; Η ΕΕ, από τον Μάιο του 2018, έχει δημιουργήσει έναν μηχανισμό για την προστασία των προσωπικών δεδομένων σε κάθε χώρα-μέλος, τον Κανονισμό Προστασίας Προσωπικών Δεδομένων, γνωστό και ως GDPR, για να αποφευχθεί η κοινοποίηση προσωπικών πληροφοριών χωρίς συναίνεση. Ωστόσο, επαρκεί ένας αυστηρός μηχανισμός για την προστασία ευαίσθητων πληροφοριών; Σαφώς όχι, γιατί οποιοσδήποτε κακόβουλος συνδυασμός δωρεάν δεδομένων από διάφορες πηγές μπορεί να στοχοποιήσει ένα άτομο, ενώ οι χάκερ μπορούν να παραβιάσουν τα συστήματα ασφαλείας εταιρειών και ιδρυμάτων και να κλέψουν τεράστιους όγκους προσωπικών δεδομένων.

Επιπλέον, η ανάπτυξη της τεχνολογίας και πιο συγκεκριμένα η πρόοδος της τεχνητής νοημοσύνης και της μηχανικής μάθησης είναι άρρηκτα συνδεδεμένη με τα μεγάλα δεδομένα και την επεξεργασία τους. Ωστόσο, τώρα οι ερευνητές και οι επιστήμονες πρέπει επίσης να εξετάσουν την προστασία δεδομένων από διάφορα κακόβουλα άτομα που προσπαθούν να τα επεξεργαστούν για τους σκοπούς τους. Λοιπόν, ποια είναι η λύση για την προστασία της ιδιωτικής ζωής;

Η απάντηση είναι η ανωνυμοποίηση δεδομένων. Η ανωνυμοποίηση προσωπικών δεδομένων είναι η διαδικασία κρυπτογράφησης ή αφαίρεσης δεδομένων προσωπικής ταυτοποίησης από σύνολα δεδομένων, έτσι ώστε το άτομο να μην μπορεί πλέον να αναγνωριστεί άμεσα ή έμμεσα. Έτσι, η ανωνυμοποίηση δεδομένων δεν περιορίζεται στην απλή κατάργηση της ταυτότητας ενός ατόμου, όπως όνομα ή επώνυμο, αλλά επικεντρώνεται στην ”απόκρυψη” δευτερευουσών πληροφοριών όπως ηλικία, ταχυδρομικός κώδικας, φύλο κ.λπ. Τα δεδομένα μπορεί να περιλαμβάνουν πληροφορίες που δεν είναι από μόνα τους μοναδικά αναγνωριστικά, αλλά μπορούν να στοχοποιήσουν οποιοδήποτε όταν συνδυάζονται με άλλα σύνολα δεδομένων. Αυτά τα κομμάτια είναι γνωστά ως οιονεί αναγνωριστικά. Για παράδειγμα, περίπου το 87 τοις εκατό του πληθυσμού των ΗΠΑ μπορεί να αναγνωριστεί μοναδικά μόνο με τον 5ψήφιο ταχυδρομικό κώδικα, το φύλο και την ημερομηνία γέννησής του μαζί. Ακόμη και σε περιπτώσεις όπου μόνο ένα μικρό ποσοστό ατόμων είναι μοναδικά αναγνωρίσιμο, μπορεί να οδηγήσει σε σοβαρή παραβίαση του απορρήτου για τα άτομα που επηρεάζονται. Δεν είναι ποτέ δυνατό να γνωρίζουμε το πλήρες σύνολο των πρόσθετων πληροφοριών που υπάρχουν εκεί έξω και, επομένως, τι θα μπορούσε να είναι ταυτοποιήσιμο;

Η ανωνυμοποίηση δεδομένων μειώνει τον κίνδυνο ακούσιας αποκάλυψης κατά την κοινή χρήση δεδομένων μεταξύ χωρών, βιομηχανιών, ακόμη και τμημάτων της ίδιας εταιρείας. Μειώνει επίσης τις πιθανότητες κλοπής ταυτότητας. Η ανωνυμοποίηση δεδομένων πραγματοποιείται με διάφορους τρόπους, συμπεριλαμβανομένης της διαγραφής, της

κρυπτογράφησης, της γενίκευσης και άλλων. Μια εταιρεία μπορεί είτε να διαγράψει προσωπικά αναγνωρίσιμες πληροφορίες από τα δεδομένα που έχει συλλέξει ή να κρυπτογραφήσει αυτές τις πληροφορίες με μια ισχυρή φράση πρόσβασης. Μπορούν επίσης να αποφασίσουν να γενικεύσουν τις πληροφορίες που συλλέγονται στη βάση δεδομένων τους.

Έχουν αναπτυχθεί διάφορα μοντέλα προστασίας δεδομένων, όπως η k-ανωνυμία, l-ποικιμολομορφία, διαφορική ιδιωτικότητα κ.λπ., τα οποία ανωνυμοποιούν τα δεδομένα ενώ περιορίζουν την απώλεια πληροφοριών. Η διαδικασία ανωνυμοποίησης πρέπει να λαμβάνει υπόψη τη χρησιμότητα των ανωνυμοποιημένων δεδομένων. Είναι σημαντικό ένα σύνολο δεδομένων να μην χάσει τη χρησιμότητά του μετά την πλήρη ανωνυμοποίηση, προκειμένου οι στατιστικολόγοι και οι αναλυτές να μπορούν να βγάλουν ασφαλή συμπεράσματα σχετικά με τα δεδομένα. Για παράδειγμα, ένας πίνακας περιέχει το ακριβές ακαθάριστο εισόδημα που κερδίζουν πέντε διευθύνοντες σύμβουλοι στον τομέα του λιανικού εμπορίου. Ας υποθέσουμε ότι τα καταγεγραμμένα εισοδήματα είναι $520.000, $230.000, $109.000, $875.000 και $124.000. Αυτές οι πληροφορίες μπορούν να γενικευθούν σε κατηγορίες όπως "< $500.000" και "≥ $500.000". Οι κατηγορίες γενίκευσης δίνονται μέσω μιας ιεραρχίας, μιας δομημένης ομαδοποίησης οντοτήτων που μοιράζονται κοινά χαρακτηριστικά. Είναι μια ισχυρή και ευρέως χρησιμοποιούμενη μέθοδος για την αναπαράσταση κοινών χαρακτηριστικών μεταξύ των οντοτήτων διατηρώντας τις διαφορές τους. Αν και τα δεδομένα είναι ασαφή, θα εξακολουθούν να είναι χρήσιμα στον χρήστη. Η αποκωδικοποίηση ανωνυμοποιημένων δεδομένων είναι δυνατή μέσω μιας διαδικασίας γνωστής ως Απο-ανωνυμοποίηση (De-anonymization ή "re-identification"). Δεδομένου ότι τα ανώνυμα δεδομένα μπορούν να αποκωδικοποιηθούν και να αποκαλυφθούν, οι επικριτές πιστεύουν ότι η ανωνυμοποίηση παρέχει μια ψευδή αίσθηση ασφάλειας. Για παράδειγμα, διαφορετικές μέθοδοι επίθεσης επαναπροσδιορίζουν έναν k-ανώνυμο πίνακα επειδή η k-ανωνυμία δεν παρέχει επαρκή προστασία έναντι της έκθεσης χαρακτηριστικών. Ωστόσο, η k—ανωνυμία είναι ένα πολύ δημοφιλές μοντέλο που διατηρεί το απόρρητο. Για να επιτευχθεί η k-ανωνυμία σε ένα σύνολο δεδομένων, πρέπει να υπάρχουν τουλάχιστον k-εγγραφές που να μοιράζονται το σύνολο των χαρακτηριστικών που θα μπορούσαν να αποκαλύψουν την ταυτότητα της οντότητας της οποίας οι ιδιότητες αντιπροσωπεύονται στο σύνολο δεδομένων. Παρόλα αυτά, η l-ποικιλομορφία ήρθε να λύσει τα προβλήματα που προέκυψαν από την k—ανωνυμία. Ο κύριος σκοπός είναι να υπάρχει ποικιλομορφία στις τιμές ευαίσθητων δεδομένων μεγαλύτερη ή ίση με l για κάθε υποομάδα που δημιουργείται από k-ανωνυμία. Ωστόσο, η l-ποικιλομορφία έχει επίσης περιορισμούς και ζητήματα που η t-εγγυήτητα βοηθά στην επίλυση. Πολλές άλλες τεχνικές ανωνυμοποίησης θα αναλυθούν σε αυτή τη διπλωματική εργασία και προσπαθούν να ανακαλύψουν άλλα προβλήματα, ωστόσο τα περισσότερα είναι ξεπερασμένα.

Από την άλλη πλευρά, υπάρχει μια σχετικά νέα και πολλά υποσχόμενη τεχνική που μειώνει απότομα την απώλεια πληροφοριών και διατηρεί το απόρρητο των πληροφοριών. Το διαφορικό απόρρητο είναι ένας μαθηματικός ορισμός του τι σημαίνει να έχεις ιδιωτικότητα και αντιμετωπίζει το παράδοξο να μην μαθαίνεις τίποτα για ένα άτομο ενώ μαθαίνεις χρήσιμες πληροφορίες για έναν πληθυσμό. Ωστόσο, υπάρχουν περιορισμοί στην ανωνυμοποίηση των μεγάλων δεδομένων. Ο τεράστιος όγκος και η πολυπλοκότητα χαρακτηρίζουν τα μεγάλα δεδομένα και είναι πολύ απαιτητικός ο χειρισμός τους στην κύρια μνήμη ενός παραδοσιακού λειτουργικού συστήματος. Τα σύγχρονα υπολογιστικά συστήματα παρέχουν την ταχύτητα, την ισχύ και την ευελιξία που απαιτούνται για γρήγορη πρόσβαση σε

τεράστιους όγκους και τύπους μεγάλων δεδομένων. Μαζί με την αξιόπιστη πρόσβαση, οι εταιρείες χρειάζονται επίσης μεθόδους για την ενοποίηση των δεδομένων, τη δημιουργία αγωγών δεδομένων, τη διασφάλιση της ποιότητας των δεδομένων, την παροχή διαχείρισης και αποθήκευσης δεδομένων και την προετοιμασία των δεδομένων για ανάλυση. Τα παραδοσιακά εργαλεία δεδομένων δεν είναι εξοπλισμένα για να χειρίζονται αυτού του είδους την πολυπλοκότητα και τον όγκο, γεγονός που οδήγησε σε μια σειρά εξειδικευμένων λύσεων λογισμικού μεγάλων δεδομένων και αρχιτεκτονικής σχεδιασμένων για τη διαχείριση του φορτίου. Η ποικιλομορφία των μεγάλων δεδομένων τα καθιστά εγγενώς πολύπλοκα, με αποτέλεσμα την ανάγκη για συστήματα ικανά να επεξεργάζονται τις διάφορες δομικές και σημασιολογικές διαφορές τους. Έτσι, οι περισσότερες λειτουργίες χειρισμού και επεξεργασίας πραγματοποιούνται στον σκληρό δίσκο, όπου οι εξειδικευμένες βάσεις δεδομένων μπορούν να αποθηκεύσουν τα δεδομένα με τρόπο που δεν απαιτεί αυστηρή τήρηση ενός συγκεκριμένου μοντέλου. Τα μεγάλα δεδομένα χρησιμοποιούνται σχεδόν σε κάθε κλάδο για τον εντοπισμό προτύπων και τάσεων, απαντήσεις σε ερωτήσεις, απόκτηση γνώσεων για τους πελάτες και αντιμετώπιση σύνθετων προβλημάτων. Οι εταιρείες και οι οργανισμοί χρησιμοποιούν τις πληροφορίες για πολλούς λόγους, όπως η ανάπτυξη των επιχειρήσεων τους, η κατανόηση των αποφάσεων των πελατών, η ενίσχυση της έρευνας, η πραγματοποίηση προβλέψεων και η στόχευση βασικών ειδών κοινού για διαφημίσεις, επομένως η ανωνυμοποίηση μεγάλων δεδομένων πρέπει να αποτρέπει τις διαρροές απορρήτου.

Στην παρούσα διπλωματική εργασία, εστιάσαμε στην ανωνυμοποίηση μεγάλων δεδομένων, εφαρμόζοντας έναν αλγόριθμο που συγκρατεί τον κύριο όγκο δεδομένων στον σκληρό δίσκο και στην κύρια μνήμη χειρίζεται βασικές πληροφορίες για αυτά. Ακολουθώντας μια εφάπαξ προσέγγιση ομαδοποίησης προκειμένου να εκτελεστούν σε κανονικούς όρους χρόνου, οι εγγραφές κατατίθενται σε συστάδες χρησιμοποιώντας μια μέτρηση ομοιότητας-απόστασης. Μετά από αυτό, τα συμπλέγματα που περιέχουν παρόμοια-πανομοιότυπα ανωνυμοποιούνται μέσω των ιεραρχιών και του μοντέλου απορρήτου k-ανωνυμίας. Αυτός είναι ένας απαιτητικός αλγόριθμος και εκτός από τον χρόνο εκτέλεσης και την απώλεια πληροφοριών, πρέπει να χειριστούμε τον διαχωρισμό των συμπλεγμάτων, τα ”μικρά” (μέγεθος μικρότερα από k) συμπλέγματα, τη γρήγορη προσπέλαση μέσω χιλιάδων συμπλ-εγμάτων κ.λπ. Η ομαδοποίηση που βασίζεται σε δίσκο. Ο αλγόριθμος αναπτύσσεται στο πλαίσιο του εργαλείου ανωνυμοποίησης Amnesia, το οποίο παρέχει πολλές τεχνικές ανων-υμοποίησης μέσω μιας φιλικής προς τον χρήστη γραφικής διεπαφής χρήστη. Τέλος, η αξιολόγηση του αλγορίθμου πραγματοποιήθηκε στο εργαλείο ανωνυμοποίησης Amnesia χρησιμοποιώντας τις μετρικές απώλειας πληροφοριών *NCP* και *Total*.

Περίληψη

Καθώς η τεχνολογία διαπερνά όλο και περισσότερες πτυχές της ζωής μας, κάθε ανθρώπινη δραστηριότητα αφήνει ένα ψηφιακό σημάδι σε κάποιο χώρο αποθήκευσης. Τεράστιοι όγκοι προσωπικών δεδομένων δημιουργούνται καθημερινά και σπάνια κάποιος γνωρίζει την έκταση των πληροφοριών που τηρούνται και υποβάλλονται σε επεξεργασία για λογαριασμό του/της. Αυτά τα προσωπικά δεδομένα εγείρουν σημαντικές ανησυχίες σχετικά με το απόρρητο των χρηστών, καθώς σημαντικές και ευαίσθητες λεπτομέρειες σχετικά με τη ζωή ενός ατόμου συλλέγονται και εκμεταλλεύονται από τρίτα πρόσωπα. Για παράδειγμα, οι ανακαλύψεις στη μηχανική μάθηση προέρχονται από τεχνικές εκμάθησης που απαιτούν μεγάλο όγκο δεδομένων εκπαίδευσης, ενώ τα ερευνητικά ιδρύματα συχνά χρησιμοποιούν και μοιράζονται δεδομένα που περιέχουν ευαίσθητες ή εμπιστευτικές πληροφορίες για διαφορετικούς ανθρώπους. Ομολογουμένως, δεδομένα υπάρχουν παντού. Ο κόσμος έχει μετατραπεί σε μια έκρηξη πληροφοριών και δεν πρέπει να αποτελεί έκπληξη, ειδικά σε μια εποχή που η αποθήκευση δεδομένων είναι φθηνή και προσβάσιμη. Ως αποτέλεσμα, η ανάγκη των εταιρειών για πληροφορίες αυξάνεται κάθε λεπτό. Οι εταιρείες πρέπει να γνωρίζουν όσο το δυνατόν περισσότερα για τους πελάτες τους. Ωστόσο, πώς μπορεί να επιτευχθεί αυτό χωρίς να διακυβεύεται η ιδιωτική ζωή των ατόμων; Πώς μπορούν οι εταιρείες να παρέχουν εξαιρετικές δυνατότητες και να διατηρήσουν το απόρρητο; Η ανάπτυξη τυπικών μοντέλων απορρήτου, όπως η k-ανωνυμία και το διαφορικό απόρρητο έχει βοηθήσει στην επίλυση αυτού του προβλήματος, έτσι υπάρχει ένας αυξανόμενος αριθμός οργανισμών που ανωνυμοποιούν δεδομένα για την προστασία ευαίσθητων πληροφοριών, όπως προσωπικές πληροφορίες, διάφορες εκδηλώσεις που συμμετέχει ένα χρήστης, πραγματική -ώρα τοποθεσίας κ.λπ. Η k-ανωνυμία είναι ένα σύνηθες μοντέλο απορρήτου που εφαρμόζεται για την προστασία προσωπικών δεδομένων των υποκειμένων σε σενάρια κοινής χρήσης δεδομένων και τις εγγυήσεις που μπορεί να παρέχει η k-ανωνυμία όταν χρησιμοποιείται για την ανωνυμοποίηση δεδομένων. Σε πολλά συστήματα διατήρησης της ιδιωτικής ζωής, ο τελικός στόχος είναι η ανωνυμία για των δεδομένων. Έτσι, μια έκδοση ενός συνόλου δεδομένων παρέχει προστασία k-ανωνυμίας, εάν οι πληροφορίες για κάθε άτομο που περιέχονται στην έκδοση δεν μπορούν να διακριθούν από τουλάχιστον άτομα k-1 των οποίων οι πληροφορίες εμφανίζονται επίσης στο σύνολο δεδομένων που κυκλοφόρησε. Η κύρια ιδέα του μοντέλου βασίζεται στην ιδέα ότι συνδυάζοντας σύνολα δεδομένων με παρόμοια χαρακτηριστικά, η αναγνώριση πληροφοριών για οποιοδήποτε από τα άτομα που συνεισφέρουν σε αυτά τα δεδομένα μπορεί να συγκαλυφθεί. Σε αυτή τη μελέτη, θα εφαρμοστεί ένας αλγόριθμος ομαδοποίησης βασισμένου σε δίσκο, ο οποίος εστιάζει στην ανωνυμοποίηση μεγάλων δεδομένων, με βάση το μοντέλο απορρήτου k-ανωνυμίας.

Λέξεις κλειδιά
ανωνυμοποίηση, k-ανωνυμία, εξόρυξη δεδομένων, ομαδοποίηση, μεγάλα δεδομένα, απόρρητο

Abstract

As technology permeates more and more aspects of our lives, every human activity leaves a digital mark on some storage space. Vast amounts of personal data are explicitly created every day, and rarely does someone know the extent of the information being held and processed for him/her. This personal data raises important concerns about users' privacy, as important and sensitive details about an individual's life are collected and exploited by third parties. For example, the discoveries in machine learning come from learning techniques that require large amounts of training data, while research institutes often use and share data that contains sensitive or confidential information about different people. Admittedly, data is everywhere. The world has become an explosion of information, and it should not come as a surprise, especially when the data storage is cheap and accessible. As a result, corporations' need for information grows by the minute. Companies need to know as much as possible about their customers.

Nonetheless, how can this be achieved without compromising the privacy of individuals? How can companies provide excellent features and maintain great privacy? The development of standard privacy models such as k-anonymity and differential privacy has helped solve this problem, so a growing number of organizations anonymize data to protect sensitive information, such as personal information, user events, and a person's real-time location, etc.

This study will implement a disk-based clustering algorithm that focuses on big data anonymization based on the k-anonymity privacy model. K-anonymity is a standard privacy model applied to protect the data subjects' privacy in data sharing scenarios and the guarantees that k-anonymity can provide when used to anonymize data. The main concept of the model is based on the idea that by combining sets of data with similar attributes, identifying information about any one of the individuals contributing to that data can be obscured. In many privacy-preserving systems, the end goal is anonymity for the data subjects. So, a dataset release provides k-anonymity protection if the information for each person contained in the release cannot be distinguished from at least k-1 individuals whose information also appears in the released dataset.

Keywords
anonymization, k-anonymity, data mining, clustering, big data, privacy

Contents

1. Introduction

In this day and age, personal data records are increasingly collected by governments, health centers, social networks, organizations, companies, and individuals for data analysis and other different purposes. Sharing and exploiting these data is the key to significant value and productivity gains, but at the same time, it poses crucial threats to user privacy. Hence, the risks of violating the user's privacy are constantly increasing especially with the rapid growth of the technology and, more specifically, the sharp evolution of the internet through which data are transmitted very quickly and are vulnerable to hackers.

The need for solid commitments and strong rules, national, European and global, for the processing of personal data is becoming more urgent as the breach of personal data can lead to serious damage to the reputation of the subjects concerned, restriction of their rights, discrimination, abuse or interception of identity, and even financial loss. On the other hand, releasing sharing and free access to data helps the scientists and gives significant opportunities to researchers to develop new methods by analyzing datasets. However, it is crucial to protect each person's information privacy in the dataset. If anybody can be explicitly distinguishable from the released data, their private data will potentially be compromised.

Consequently, it is very essential that before someone releases a dataset, the private data must be secured so that the identity of a person contained in the information cannot be discerned and at the same time, the whole data should be still useful and processable. Therefore, it is necessary to develop efficient techniques and methods for data anonymization, for which the main challenge is to guarantee a small trade-off between the efficacy of the privacy guarantee and the quality of the anonymous data offered.

1.1 Privacy Leakage

As mentioned above, the huge volume of collected data and the many ways of sharing has led to a rapidly increasing accumulation of personal data, leading to privacy issues such as the reporting of sensitive data and the massive collection of personal information by third parties. [1].

The anonymization procedure is not limited to removing direct identifiers that might exist in a dataset, e.g. the name or the person's Social Security Number; it also includes removing secondary information, e.g. age, zip code that might lead indirectly to the true identity of an individual. This form of identification called linking attack, many studies show that the majority of the U.S. population can be uniquely distinguished by linking zip code, gender, and date of birth, making it clear that published information containing these characteristics can not be considered anonymous data [1–3]. This secondary information is referred to as quasi-identifiers which will be analyzed in detail in the following chapters.

Consider the following example to better understand how secondary information can be used to re-identify a person, consider the following example. A publisher that owns pa-

tients' medical data wants to publish an anonymized version of the data. The dataset is superficially anonymized by removing direct identifiers e.g., names and social security numbers, but descriptive information like the zip code of the patient's residence and her/his age remain. An adversary who wants to identify the patients that are related to the anonymized data, may have access to such descriptive information from other sources, e.g., a voter's registry. The re-identification can be achieved by matching the descriptive information (zip code, age) of the anonymized data to the public registry. If a single match is produced for a given combination, then a patient can be accurately identified as depicted in Figure 1.

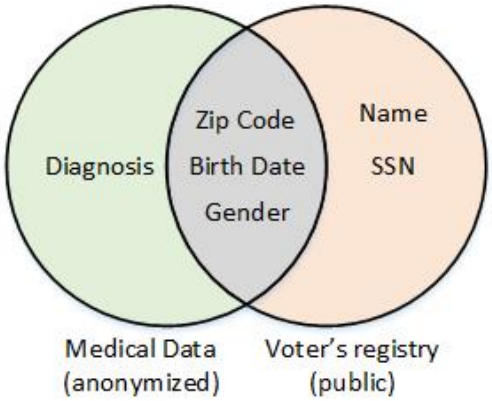

Figure 1: Data Linking

The sparser the data are, the more unique combinations exist, and the easier it is for an adversary to locate unique records that correspond to specific users. Below, it is examined a few instances of privacy protection failures that led to identifying individuals or users.

1.1.1 Netflix Competition

A typical and well-known example is the Netflix Crowdsourcing competition which began on October 2, 2006. Netflix the largest online video streaming service, provided a training dataset with over 100 million ratings which approximately 480 thousands users gave to over 17 thousands movies. Moreover, all the direct personal information such as names, surnames etc. had been removed from the dataset which contained only an anonymized user Id, ratings, and the dates the subscriber rated the movie. However, Narayanan and Shmatikov [4] from the University of Texas demonstrated that an attacker who knows little about individuals contributing could recognize the subscriber information in the dataset. By combining the IMDB as background knowledge with the Netflix's open dataset they achieved to identify subscribers' record and revealed personal sensitive information [5].

1.1.2 Group Insurance Commission (GIC)

As described and presented in Figure 1 data linking is a serious problem and everyone is capable to match secondary information in order to identify a specific person, Latanya

Sweeney shows us how simple is the above procedure. In Massachusetts, the Group Insurance Commission (GIC) is responsible for purchasing health insurance for state employees. GIC collected patient-specific data with nearly one hundred attributes such as diagnosis, medication etc. per encounter for approximately 135,000 state employees and their families. Because the data were believed to be anonymous, GIC gave a copy of the data to researchers and sold a copy to industry. Latanya Sweeney purchased the voter registration list and she managed to link it with the (GIC) data by using zip code, birth date, gender and identified diagnosis, procedures, and medications to particularly named individuals. For example, William Weld was governor of Massachusetts at that time and his medical records were in the (GIC) data. Governor Weld lived in Cambridge Massachusetts. According to the Cambridge Voter list, six people had his particular birth date; only three of them were men; and, he was the only one in his 5-digit ZIP code [1]. The example above provides a demonstration of re-identification by directly linking (or "matching") on shared attributes which provide secondary information (quasi-identifiers) about a person.

1.1.3 American Online (AOL) Search Log

Search engine's logs are a vast source of information for both researchers and marketing companies, but at the same time, their publication may expose the privacy of the users from whom the logs are created. There is at least one case of such exposure of users' personal search logs where very simple anonymization procedure was applied, revealing much information for their identification. Another shocking example is the release of 20 million detailed search logs of many numbers of AOL users collected over a three months' period for research purposes. AOL made the in an effort to help the "Information Retrieval Research Community" which has dealt a severe blow both the privacy policy of AOL's users as well as AOL itself, with lawsuits and objections against it. Ideally search logs should be anonymous before their publication. The problem is that achieving the desired level of privacy in logs becomes difficult, while balancing between usability and data privacy. There are several approaches to anonymize such data, but they are usually limited to deleting specific queries or logs. In addition, common techniques used in disclosure statistical analysis have never been applied, until recently for such cases.

1.1.4 Cambridge Analytica and Facebook

In 2013, University of Cambridge psychology professor Dr. Aleksandr Kogan created an application called "thisisyourdigitallife." This app, offered on Facebook, provided users with a personality quiz. After a Facebook user downloads the app, it would start collecting that person's personal information such as profile information and Facebook activity (e.g., what content was "liked"). Around 300,000 people downloaded the app. But the data collection did not stop there. Because the app also collected information about those users' friends, who had their privacy settings set to allow it, the app collected data from about 87 million people. Then, Dr. Kogan passed this data on to Strategic Communication Laboratories (SCL), which owns Cambridge Analytica (CA), a political consulting firm that uses data to

determine voter personality traits and behavior. It then uses this data to help conservative campaigns target online advertisements and messaging. The scandal of Facebook and Cambridge Analytica indicates us the need to impose a tough legal framework that will force companies to anonymize data and the further extension of the existing anonymization techniques.

1.2 Privacy Protection

In a study by Paul Ohm [6], a professor of Law at the Georgetown University Law Center in Washington, points out that for almost every human being on earth there is at least one digital information that, implies a fact of his/her life, stored in a database. A malicious agent can use this digital record to harm the victim, whether it is through blackmail, harassment, or even identity theft.

The privacy protection field deals with developing various algorithms and anonymization techniques, which remove identifying information from the provided data so that the attacker can not identify individuals. A common technique used in anonymization models is that of generalization [7]. Generalization is when the initial value that displayed in data is replaced by a more general value, such as a set of values that may contain it. The set of possible generalizations of the values, forms the generalization hierarchy. A part of the initial information is lost with the generalization technique.

For this reason, privacy guarantees are sought that prevent the transfer of personal information but while removing as little information as possible from the original dataset while maintaining their usefulness for those who want to use it. The basic guarantee which mentioned above that satisfactorily prevents the identification of a record is k-anonymity [2] . The purpose of k-anonymity privacy-model is that no value combination should appear in the database in less than k-records.

In addition, another powerful privacy-model is Differential Privacy [8]. Differential Privacy offers a strict privacy guarantee with particularly favorable, powerful properties and strong mathematical definition for preserving the privacy of individuals. The general principle of Differential Privacy is that the attacker can learn virtually nothing more about an individual than they would learn if that person's record were absent from the dataset. The guarantee is strong enough because it aligns with real world incentives—individuals have no incentive not to participate in a dataset, because the analysts of that dataset will draw the same conclusions about that individual whether the individual includes himself in the dataset or not. As their sensitive personal information is almost irrelevant in the system's outputs, users can be assured that the organization handling their data is not violating their privacy.

1.3 Big Data Anonymization

Our age is characterized by the way big data is processed in many sectors such as security, extracting important results and statistics, the storage of them etc [9]. For quite some time, organizations and companies have been investing in developing solutions for implementing systems with enough computing power to process large volumes of data. However, big data are so large and complex containing so much volume that can not be handle by the main memory of an operation system and generally by traditional data-processing application softwares. The analysis of this data is very important for both businesses and public-interest organizations. Data with many fields required greater statistical power, while data with higher complexity may lead to a higher false discovery rate [10].

Indeed, the complexity is quite high, while at the same time, the data types are many. For instance, some data is stored in a traditional relational database, but data such as documents, various files as well as videos and images are unstructured. New data sources have emerged in recent years, such as data generated by social networks or applications on smart devices. Therefore, as mentioned above, since the technology has penetrated very deeply into the society [11] and most of the big data concerns ordinary individuals, so, it is imperative to anonymize facing their complex management. However, it is impossible for algorithms and software tools that use only a computer's main memory to manage big data which contain information in different formats or even to anonymize them.

In this thesis we designed and developed an algorithm which is customized to big data and utilizes the hard disk so that the main memory is not overloaded. Using a clustering logic [12] we split the data into clusters in order to load the data in small quantities into the main memory which provides the anonymization procedure via k-anonymity guarantee using hierarchies [13]. In a nutshell, our method stores the dataset in an embedded database on a hard disk then, divides the data into clusters and finally, applies the k-anonymity privacy-model to these clusters.

1.4 Amnesia Anonymization tool

The disk-based clustering algorithm has been implemented in the context of developing and upgrading the Amnesia Anonymization Tool. Amnesia is a flexible, user-friendly, free, and open-source data anonymization tool. More specifically, Amnesia converts relational and transactional data into anonymous datasets to which formal privacy guarantees apply by [14] removing immediate identifiers (names, SSNs, etc.) and converting secondary identifiers [1] (dates of birth, postal codes, etc.) while focusing on applicability and user-friendliness. Amnesia provides an easy-to-use editing mode, a module for auto-generating hierarchies, efficient and scalable algorithms for anonymizing relational and transactional datasets through k-anonymity [2,7] and k^m-anonymity [15], Users can regulate the information loss and guide the anonymization process by graphically exploring candidate solutions, observing data mining statistics, and mixing user-defined suppression rules with algorithm-

[1] https://privacy-analytics.com/

generated generalization-based solutions.

1.5 Thesis Structure

This section provides a summary of the following chapters and the organization of the current thesis, giving better guidance and understanding to the reader.

- In **Chapter 2**, we analyze the main anonymization techniques and privacy models dividing them into two categories, generalization and randomization. The former includes many classic privacy techniques that have been applied for several years such as k-anonymity, l-diversity, t-Closeness, δ-Presence and k^m-Anonymity and the corresponding algorithms that apply the aforementioned techniques. Moreover, we presente many examples that show how the algorithms modify the data to anonymize them as well as the disadvantages and advantages of the them. Finally, we define and formalize the promising notion of differential privacy providing basic definitions and theorems while showing known mechanisms and techniques for achieving differential privacy.
- In **Chapter 3**, we present in detail the proposed algorithm for anonymizing big data, which achieves less information loss by applying local recording and employing a clustering method storing the main volume of the data in the hard disk. Initially, we described related clustering approaches presenting our new corresponding clustering method. Moreover, we analyze the basic methods and describe the main data structures. Furthermore, we list the problems and necessary optimizations, and we evaluate the algorithm using graphs regarding execution time and information loss. Finally, we present a use case scenario utilizing the Amnesia Anonymization Tool.
- In **Chapter 4**, we summarize and conclude the current study proposing future work for the disk clustering algorithm and possible extensions.

2. Anonymization Techniques

As described previously, increasing access to vast data is vital for reasearch and study puproses mainly of scientific, economic and statistical interest. However, it poses serious risks concerning the privacy of individuals. For personal data to be analyzed without the risk of losing privacy, various anonymization techniques have been developed, varying from simple to very complex ones.

Anonymization is a special category of processing personal data that aims to prevent the verification of the identity of the data subject but, at the same time, the data must remain valuable. There are essentially two different approaches of data anonymization. The former is based on the randomization of data and the latter on the generalization of the specifics of a database as noted below in this chapter.

In order to better understand the study of the problem of privacy protection, it is advisable at this point to clarify some useful definitions that are often used in the scientific community.

- **Personal data:** The set of personal information of an individual such as his/her gender, job position, age or salary which determine him/her. These data are usually collected in databases, so that they can be more easily processed and transferred.
- **Table:** Data which organized in a tabular format $(A_1, A_2, ..., A_n)$ and are part of a relation database where $A_1, A_2, .., A_n$ are its attribute columns.
- **Attribute:** Defines a specific view of the sample objects and is displayed as a separate column in the data table having a set of possible values.
- **Key attributes or Direct identifiers:** They directly identify a person such as full name, ID, Social Security Number etc. which are removed from the data table before being published.
- **Sensitive attributes:** It is a subset of table's attributes that refers to personal information which is required to remain confidential as they relate to sensitive personal data.
- **Quasi-identifiers:** It is a subset of $Q_I = (A_i, .., A_j)$ (where $1 <= i <= j <= n$) attributes of the table $T = (A_1, A_2, .., A_n)$ which can lead to the recognition of an individual's identity if they link or combine with other data.

2.1 Pseudo-anonymization

Pseudonymization is a data management and de-identification procedure by which personally identifiable information fields (names, ids, phone numbers, etc.) are removed or replaced by one or more artificial identifiers, or pseudonyms from a sensitive dataset (health records, medical prescriptions, financial information, online surveys, workplace files, etc.) [16]. A single pseudonym for each replaced field or collection of replaced [17] fields makes the data record less identifiable while remaining suitable for data analysis and data processing. Pseudonymized data can be restored to its original state with the addition of information which allows individuals to be re-identified. In contrast, anonymization is intended to prevent re-identification of individuals within the dataset.

2.1.1 Masking

Masking is a pseudo-anonymization method that refers to hiding some of the information in the dataset using alternative characters. Masking techniques are widely used for hiding parts of credit card numbers during credit card processes and payments. However, remaining identifying data (date of birth, zip code, gender, marital status, etc.) could be combined

to re-identify persons and compromise their privacy. Note that the triple (date of birth, gender, zip code) is enough to uniquely identify at least 87% of US citizens in publicly available datasets.

2.2 Generalization

The process that more general values replace quasi-identifiers while, the purpose is to preserve part of the original value information without changing it completely. In turn, this generalized value can be generalized again to a more generalized value, again maintaining the same semantics as the initial field value, and so on.

2.2.1 Generalization Hierarchy

All generalization levels at which a database value can be generalized constitute a generalization hierarchy [18] which are usually represented in a tree form. The generalization technique applies to both numerical and categorical data. In the former, the initial value is replaced by a value interval containing the initial value, which can be further generalized to a larger value interval and so on. A typical example of numerical data generalization is presented below in figure 2 and concerns the annual income feature, where the initial value that reflects the income of a person can be generalized over an interval of five thousands dollars, twenty thousands dollars, etc. Concerning the latter, each value field is generalized to a more general value based on the semantics of the initial values, as shown in the figure below 3. There are two ways to apply the Generalization technique in a dataset, Global Generalization (Global Recording) and Local Generalization (Local Recording).

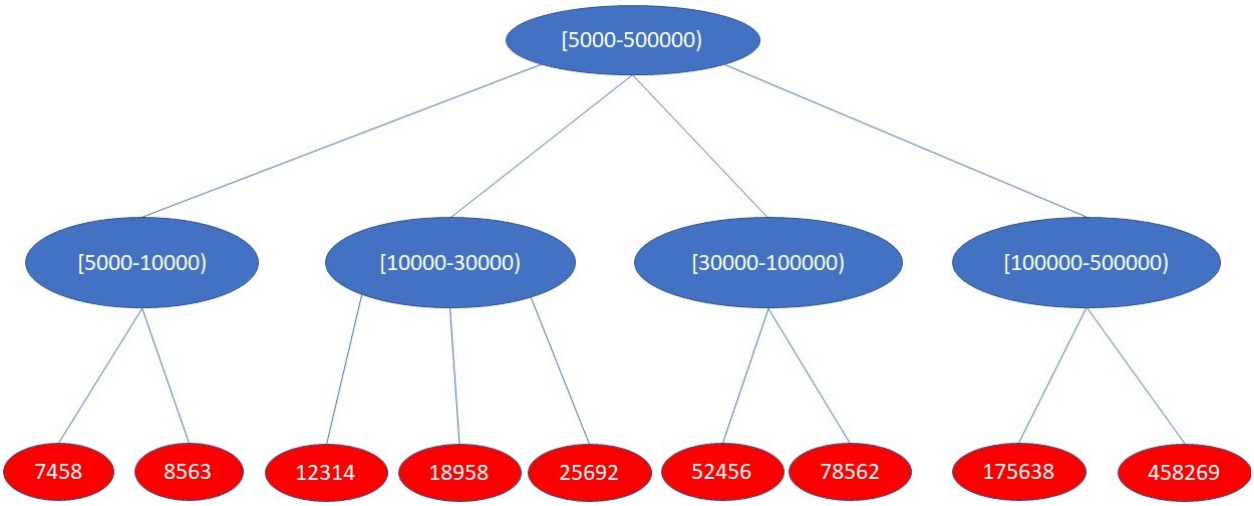

Figure 2: Hierarchy for numerical

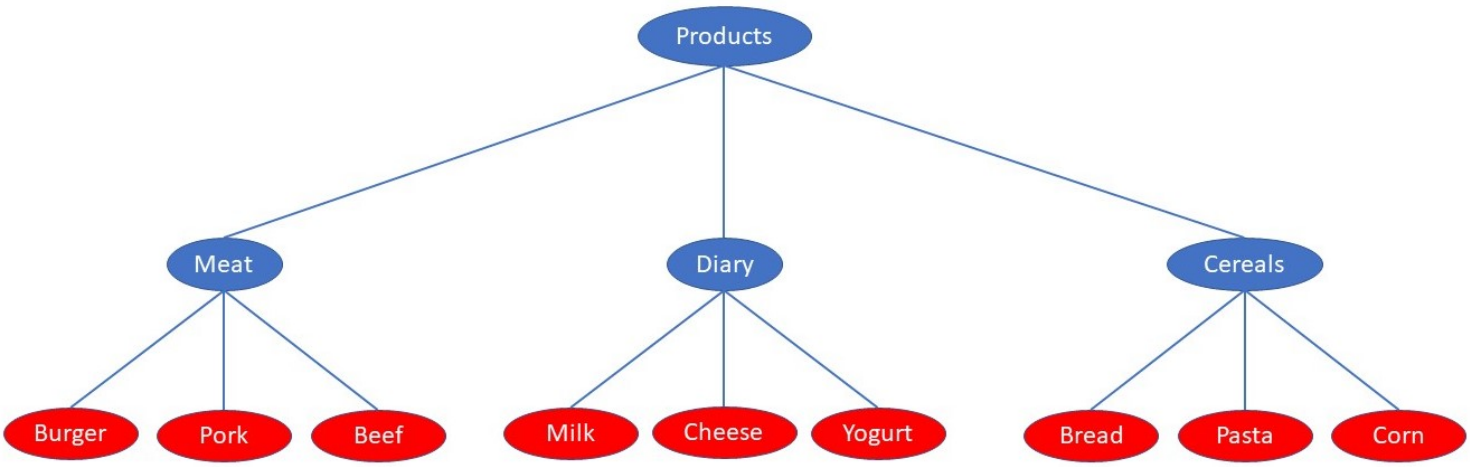

Figure 3: Hierarchy for categorical

2.2.2 Global Generalization

Global Generalisation was proposed by Samarati and Sweeney [2, 19]and maps the entire domain of each quasi-identifier attribute in T to a more general domain in its domain generalization hierarchy. This scheme guarantees that all values of a particular attribute in V belong to the same domain. In this case, the anonymized datasets have all the attributes equally generalized in all the entries. If different records had the same attribute value, this value would correspond to the same generalized value as depicted in image 4.

Id	Age	Zipcode	Diagnosis
1	28	13053	Heart Disease
2	29	13068	Heart Disease
3	21	13068	Viral Inflection
4	23	13053	Viral Inflection
5	50	14853	Cancer
6	55	14853	Heart Disease
7	47	14850	Viral Inflection
8	49	14850	Viral Inflection
9	31	13053	Cancer
10	37	13053	Cancer
11	36	13222	Cancer
12	35	13068	Cancer

Global
Generalization

Id	Age	Zipcode	Diagnosis
1	[0-100]	*****	Heart Disease
2	[0-100]	*****	Heart Disease
3	[0-100]	*****	Viral Inflection
4	[0-100]	*****	Viral Inflection
5	[0-100]	*****	Cancer
6	[0-100]	*****	Heart Disease
7	[0-100]	*****	Viral Inflection
8	[0-100]	*****	Viral Inflection
9	[0-100]	*****	Cancer
10	[0-100]	*****	Cancer
11	[0-100]	*****	Cancer
12	[0-100]	*****	Cancer

Figure 4: Global Generalization dataset (Initial -> Generalized)

Global Generalisation can be described as single-dimensional generalization or multidimensional generalization. A single-dimension recoding defines some function $\phi_i : D_{Q_i} \rightarrow D'$ for each attribute Q_i of the quasi-identifier. A generalization V of T is obtained by applying each ϕ_i to the values of Q_i in each record of T. Whereas, multidimensional generalization is defined by a single function $\phi : \langle D_{Q_1} \times \ldots \times D_{Q_n} \rangle \rightarrow D'$ which is used to recode the do-

main of value vectors associated with the set of quasi-identifier attributes. Generalization V of T is obtained by applying φ to the vector of quasi-identifier values in each record of T [20].

Global Generalisation achieves anonymity by mapping the quasi-identifier attribute domains to general values. The advantage of the method is that the anonymized table will have a homogeneous set of values while its disadvantage is that the original dataset is overgeneralized and as a result the information loss is too large.

2.2.3 Local Generalization

In Local Generalization two records with same quasi-identifier value, may have different generalized values. In contrast to Global, Local generalization hides attributes per entry element. The result is a set of data with less generalised data than Global Generalization. The data space is divided into different regions and all records in the same regions are assigned to the same generalized group. Since the quality limits are quite large, it is not easy to prove the best solution given to generalization [21]. Local generalization models capture (indistinguishable) individual data items in generalized values 5. Moreover, Local Generalization produces much better results in a real-life scenario and is much more useful than Global Generalisation.

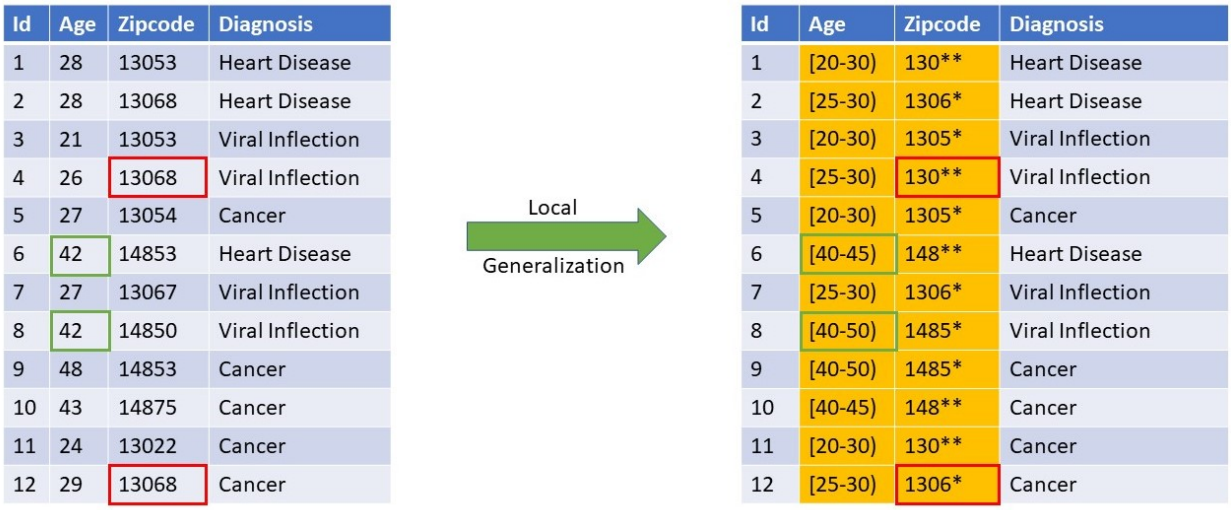

Figure 5: Local Generalization dataset (Initial -> Generalized)

2.3 Suppression

Like generalization, suppression can be applied to both cells and entire attributes. It is recommended to remove sensitive attributes as well as records to reduce the amount of generalization when k-anonymity [1,2] is achieved. The combination of generalization and

suppression was used to create different algorithms to satisfy k-anonymity which we will analyze in the next section.

The conventional framework of such an algorithm always starts with hiding many sensitive attributes and then splitting records of the remaining features into groups as it replaces the exact values of the quasi-identifiers with their generalized forms for each group, called equivalence classes. This generalization is a homogeneous [22, 23].

2.4 k-Anonymity

Researchers have proposed various methods to preserve privacy to deal with the shortcomings of simple data anonymization. One of the most popular methods of privacy preservation is the k-anonymity. For k-anonymity to be achieved in a data set, there need to be at least k-records that share the set of attributes that might reveal the identity of the entity whose properties are represented in the dataset [1]. In other words, a dataset is said to be k-anonymous if every combination of values of the quasi-identifiers in the data set appears at least in k different records. K-anonymity is defined as the level of data protection on inference by linking. It prevents linking the released data to other information sources (background information) [7].

The k-anonymity technique aims to transform the data of a table in such a way that each of the records becomes indistinguishable from other $k-1$ records in the table. Therefore, k-anonymity can be considered as an application of the grouping technique where data is divided into k groups. So, the main goal of this privacy method is to reduce the possibility of identifying an individual, even by intersection of published records that may be anonymous. In order to do this, there must be a maximum probability at most $1/k$ and an attacker being able to find out at data linking, to whom the record belongs.

Through the generalization method, the values of the quasi-identifiers fields are converted to a more general form to create equivalence classes. As explained above, generalization hierarchies are used to generalise the quasi-identifiers attributes in order to satisfy k-anonymity privacy model as we can observe in the figures below 6, 7, 8.

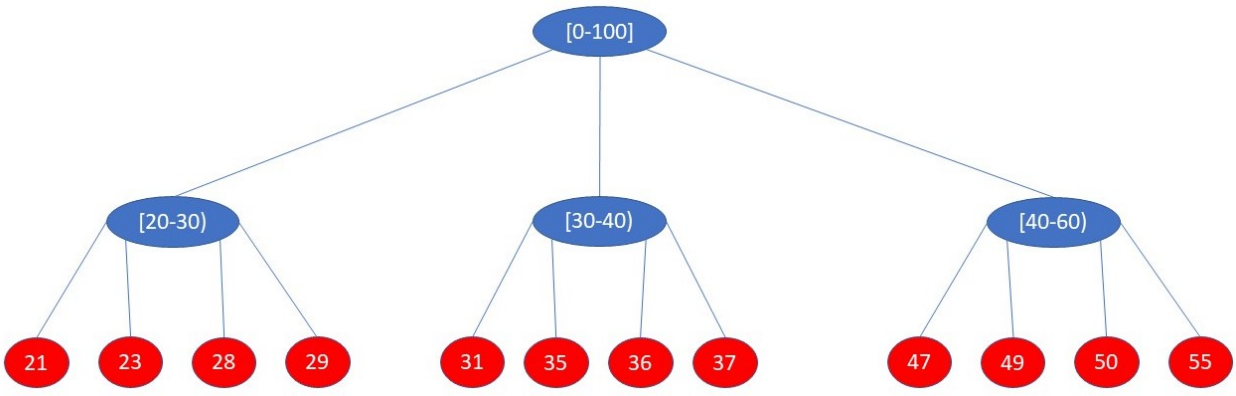

Figure 6: Generalization Hierarchy for 'Age' quasi-identifier

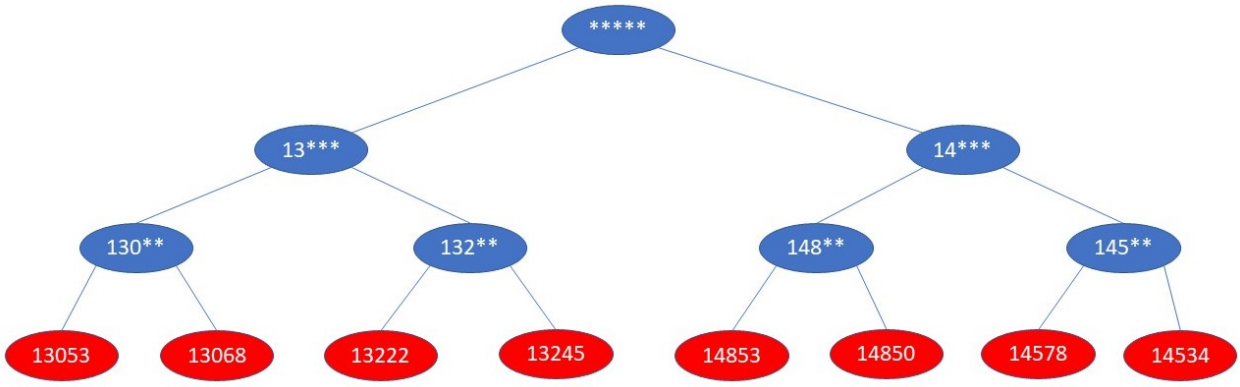

Figure 7: Generalization Hierarchy for 'Zipcode' quasi-identifier

Id	Age	Zipcode	Diagnosis
1	28	13053	Heart Disease
2	29	13068	Heart Disease
3	21	13068	Viral Inflection
4	23	13053	Viral Inflection
5	50	14853	Cancer
6	55	14853	Heart Disease
7	47	14850	Viral Inflection
8	49	14850	Viral Inflection
9	31	13053	Cancer
10	37	13053	Cancer
11	36	13222	Cancer
12	35	13068	Cancer

4-Anonymity

Id	Age	Zipcode	Diagnosis
1	[20-30)	130**	Heart Disease
2	[20-30)	130**	Heart Disease
3	[20-30)	130**	Viral Inflection
4	[20-30)	130**	Viral Inflection
5	[40-60)	148**	Cancer
6	[40-60)	148**	Heart Disease
7	[40-60)	148**	Viral Inflection
8	[40-60)	148**	Viral Inflection
9	[30-40)	13***	Cancer
10	[30-40)	13***	Cancer
11	[30-40)	13***	Cancer
12	[30-40)	13***	Cancer

Figure 8: Generalization to satisfy *4-Anonymity*

Furthermore, k-anonymity is the combination of generalization and suppression [2]. It is a harmonious combination of computational complexity and quality of the anonymized data. More specifically, the most common combination used by most k-anonymity application algorithms is the generalization at the column level and suppression at the row level of a data table. In addition, suppression is applied in direct identifiers in order to remove them from the data table before being published. An example of suppression is depicted in the figure 9.

To sum up, the parameters below, describe the problem of k-anonymity and are competitive for the best trade-off between generalization and information loss:

- **Generalization:** The amount of information that lost by generalizing the data to some generalization level ,the higher in the generalization hierarchy the greater the information loss.
- **Suppression:** The number of records removed from the data, in the anonymization procedure.

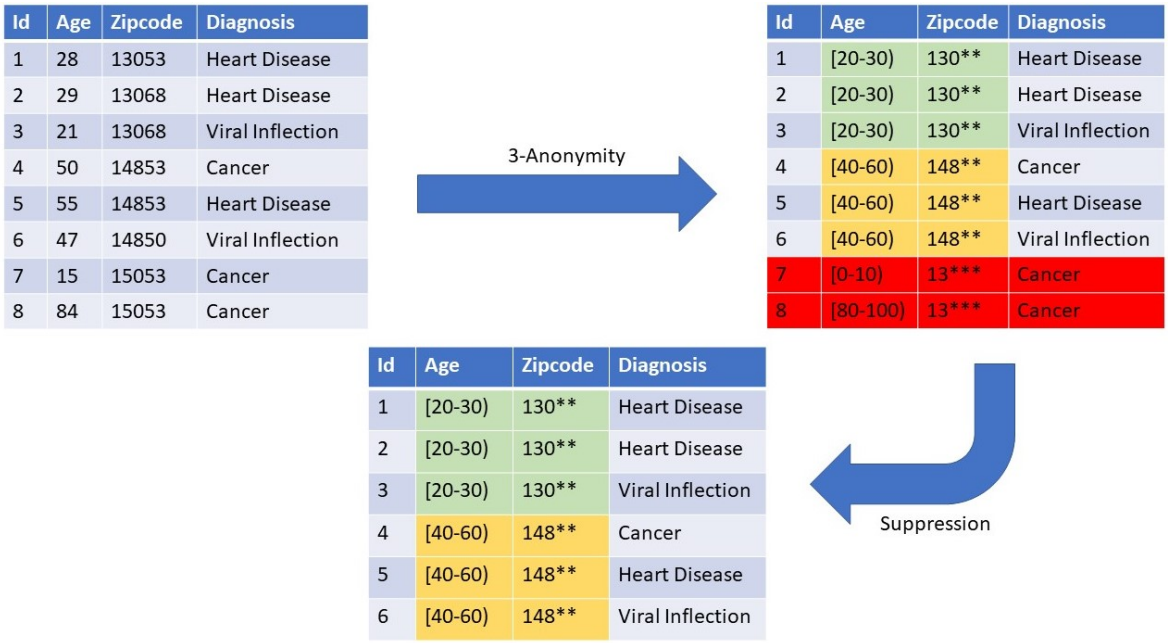

Figure 9: Generalisation + Suppression

Id	Surname	Age	Gender	Post – Code
1	Smith	44	Male	12278
2	Anderson	57	Male	23448
3	Garcia	53	Female	31284
4	Johnson	45	Male	45026
5	Miller	49	Female	45354
6	Williams	42	Male	12456
7	Brown	59	Male	23781
8	Taylor	54	Female	31695

Table 1: Example Dataset

- **Anonymity:** the minimum tolerable size of k for each equivalence class.

2.4.1 Incognito Algorithm

Incognito is a full-domain generalization algorithm that uses the approach of dynamic programming with the help of subset property. A relation T onset of attributes is said to be k-anonymous with respect to a chosen set of attributes Q if all the subsets P of the set of attributes are k-anonymous $P \subseteq Q$ [20]. Using the predefined generalization hierarchy creates a multi-attribute generalization lattice. The lattice illustrates schematically all the possible combinations between the levels of the quasi-identifiers generalization hierarchies, where all possible generalizations of the records are expressed. The algorithm aims to find the minimum global generalization, so that there is the least possible information loss.

The whole procedure of the algorithm can be enumerated in 3 simple steps using the dataset in table 1:

- **Step 1:** In the first step, the domain and value generalization hierarchy is defined for all the quasi-identifiers. Figures 10, 11, 12 show the possible domain generalization for the attributes *post-code* , *age* and *gender*.
- **Step 2:** If each of the quasi-identifiers has distinct domains, the domain generalization hierarchy formed in step one can be combined to form a multi-attribute generalization lattice. Figure 13 shows the generalization lattice created for the quasi-identifiers. Each node in the lattice represents a generalization solution. In the lattice shown in Figure 13, the node <P0, G1, A1> is a direct generalization of <P0, G1, A0> and is an implied generalization of <P0, G0, A0>.
- **Step 3:** The third step is to perform the anonymization of data. Using a breadth-first search algorithm, the lattice is traversed. While traversing the lattice, each node is checked to see if k-anonymity is satisfied. If a node satisfies k-anonymity then all its direct generalizations are removed as it is guaranteed that the subsets also satisfy k-anonymity.

The algorithm's complexity is ultimately exponential in terms of the size of the set of quasi-identifier features. It is a correct and complete algorithm for the k-anonymization that is produced, but the main disadvantage of Incognito is the generation of all possible anonymizations of the dataset from which only the most efficient one is used, so the algorithm is time consuming.

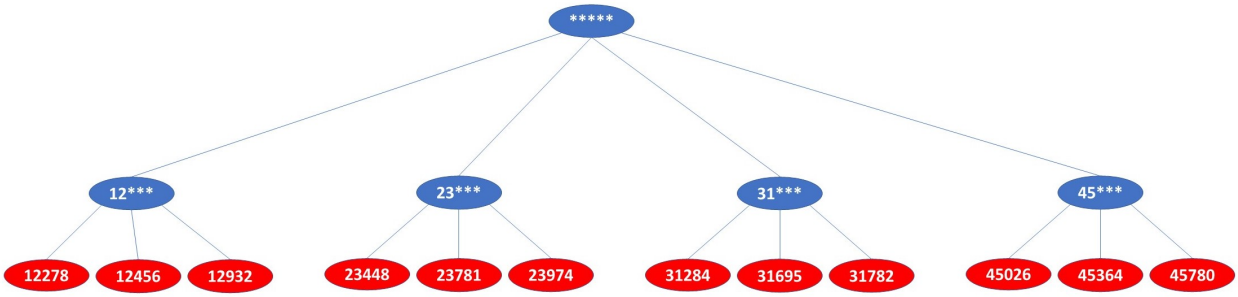

Figure 10: Generalization Hierarchy of post-code

2.4.2 Flash Algorithm

This anonymization algorithm uses the same concept of the lattice as Incognito in figure 13 and it goes over the lattice using a bottom-up breadth-first approach [24]. Flash uses a greedy depth-first strategy and the lattice is traversed vertically. It uses predictive tagging to reduce the number of nodes to be examined. Flash algorithm iterates through every node and finds the path from that node to the next node that only has tagged successors. If that condition is not fulfilled, the path will be from the node in question to the top node. The

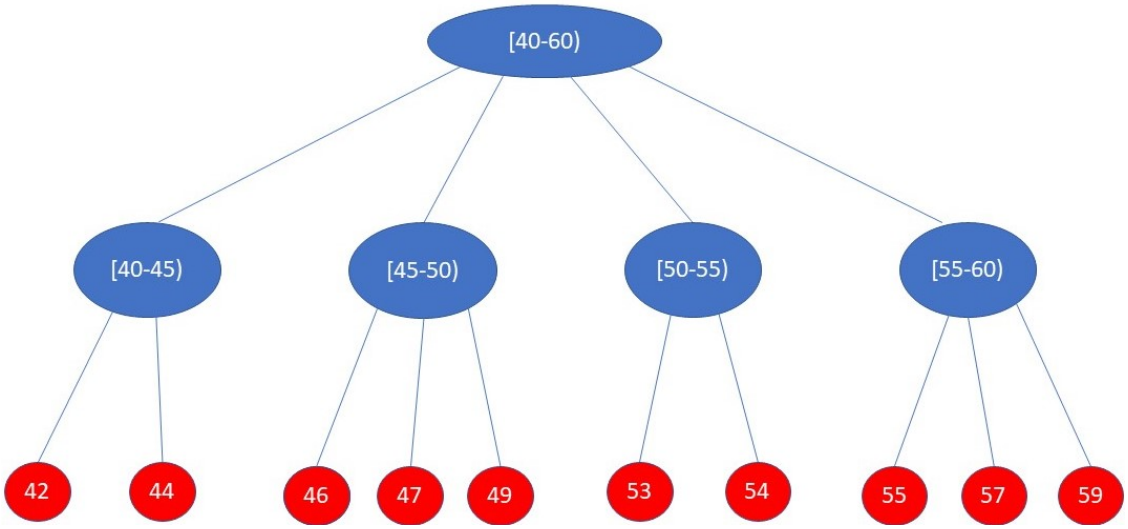

Figure 11: Generalization Hierarchy of age

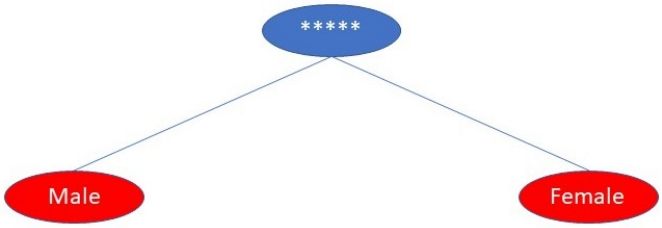

Figure 12: Generalization Hierarchy of gender

created path is checked using binary search so that anonymous nodes are tagged and non anonymous are added to the heap. After the path check is done, the algorithm continues to next iteration using the heap nodes. The whole procedure is explained in detail as an illustration below.

Suppose we have three attributes: Birth place, Birth year and Zip code for natural persons, which are matched to the respective hierarchies as shown in figures 14 15 16. We can have a solution where countries are generalized to continents (level 1 in the first hierarchy), birth dates have not been generalized (level 0 in the second hierarchy) and zip codes have their last digit removed (level 1) in the third and last hierarchy. This solution represents a node in the generalization lattice, which we mark with [1,0,1], denoting this way the generalization level of each attribute. The lattice is ordered as in Figure 17, where the nodes of each horizontal level, have the same number of total generalization levels.

Flash, as mentioned above, uses a depth first strategy, which allows it to prune several solutions and avoid examining them.Flash builds a path towards the top node, implementing a greedy depth-first strategy for every node in each level, if the node is not already tagged (as anonymous or not anonymous). The construction of a path is based on a vertical traversal strategy aiming at choosing nodes according to three fixed criteria:

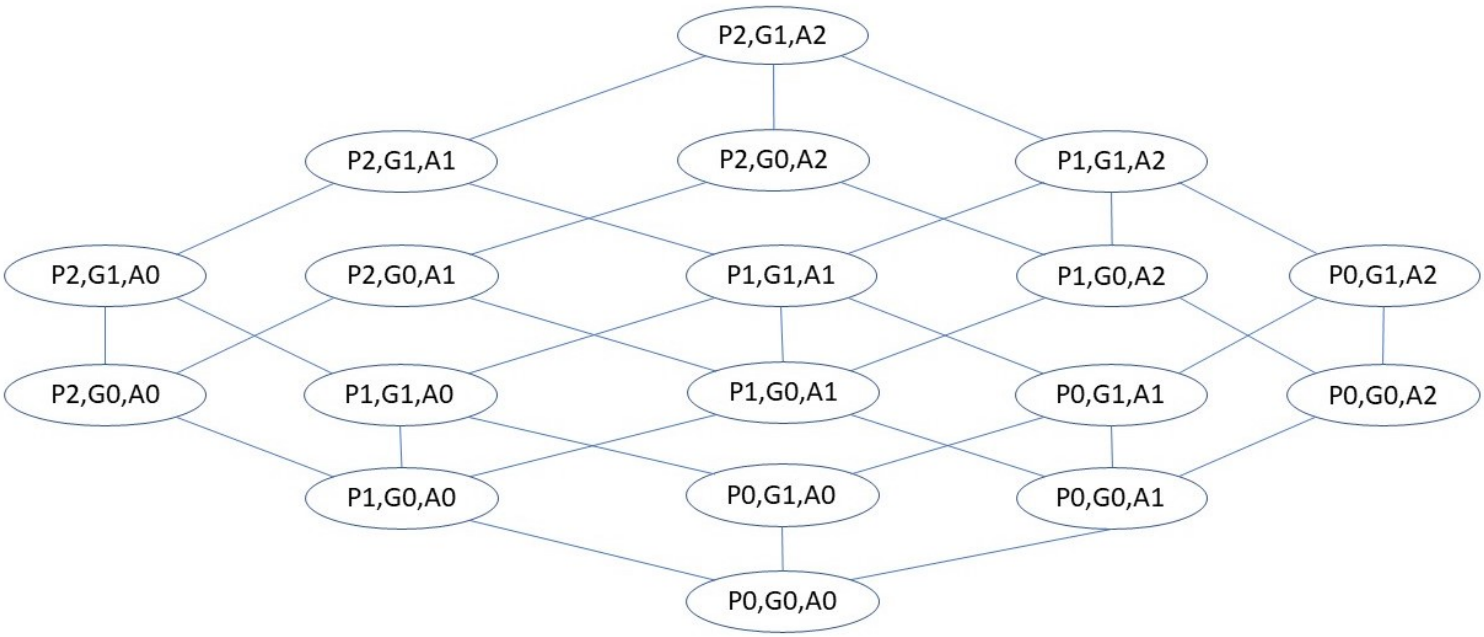

Figure 13: Generalization Lattice of quasi-identifiers

1. The total generalization level of the node in the lattice.
2. The average generalization of all quasi-identifiers of the node.
3. The average of the number of distinct values on the current level of each quasi-identifier.

The search is terminated when the top node is reached or the current node does not have a successor that is not already tagged.

When a path is built, the algorithm starts checking for k-anonymity with a binary search strategy. It starts with the node at the middle of the path, then continues with the path to the bottom of the lattice or to the top, depending on whether the middle node was anonymous or not. Whenever a node is checked, predictive tagging is applied within the whole generalization lattice. This allows completely avoiding to examine other nodes. For instance, if a node is not anonymous, then all nodes, in all paths from the bottom of the lattice to that node, will not be anonymous either. The algorithm continues until all nodes in the generalization lattice are checked for anonymity.

Figure 17 depicts the first iteration of the Flash algorithm. A path is constructed from root node [0,0,0] to reach top node [2,2,3]. This path contains nodes linked with red arrows in the figure. Then, Flash checks for 2-anonymity node [1,0,3] which is the mid-node of the path. As this node is not 2-anonymous, all predecessors of this node are also tagged non-anonymous and the algorithm continues by examining the upper half path containing nodes [1,0,3], [2,0,3], [2,1,3] and [2,2,3]. Again mid-node [2,1,3] is checked which is 2-anonymous so successor [2,1,3] is also tagged as anonymous and predecessor [2,0,3] is checked for anonymity, which does not hold. Finally, all nodes of the path have been checked and then the algorithm proceeds with the same process for the nodes of level 1. Two of them have

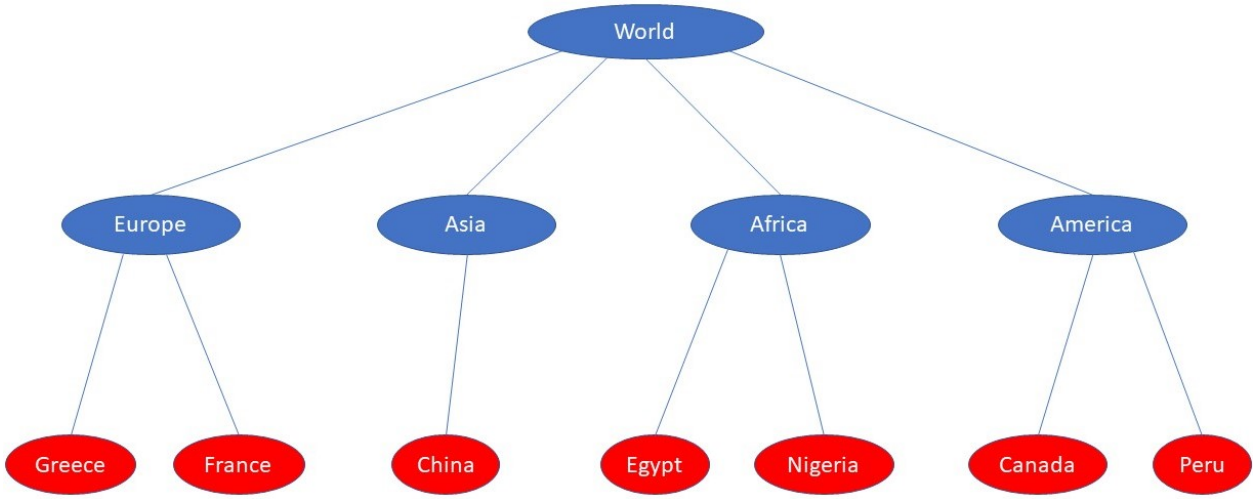

Figure 14: Generalization Hierarchy of birth-place

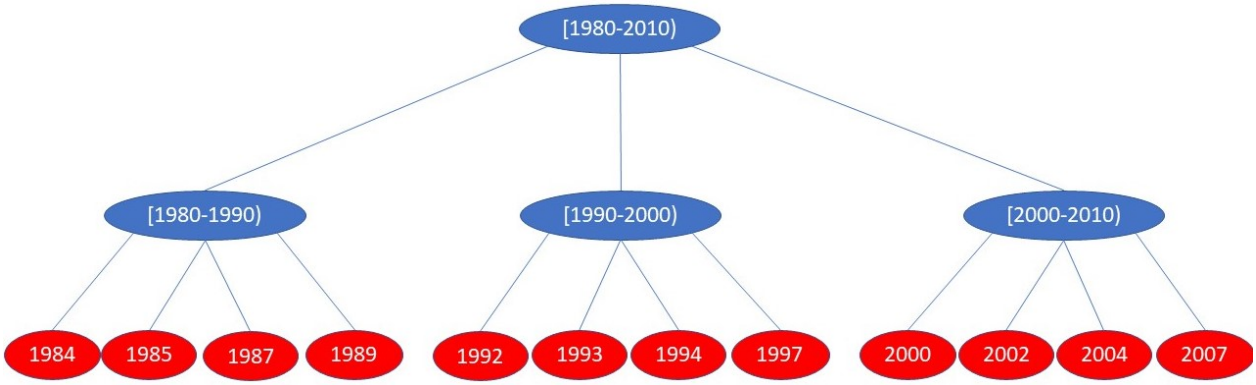

Figure 15: Generalization Hierarchy of birth-year

been tagged from the previous stage so only node [0,1,0] is a candidate, from which Flash will construct a new path towards the top node and traverse it in the same way. This process continues until all nodes of the lattice are checked. The traversal strategy employed by Flash gives a clear advantage over Incognito's breadth-first traversal.

2.4.3 Mondrian Algorithm

As already mentioned, the above algorithms apply global generalization to quasi-identifiers. The drawback of this approach is the over-generalization of the data which become useless. For example, in a numerical database, a full-domain generalization means the replacing of all the initial values with fixed separate intervals or their completely concealment.

These kind of problems are solved by the Mondrian [25] algorithm, offering a higher quality anonymization, due to the multidimensional local recording model with which it can be applied. Based on this model, a space of -dimensions is defined, where is the number of quasi-identifiers. By dividing this space into partitions, a k-anonymous solution is sought.

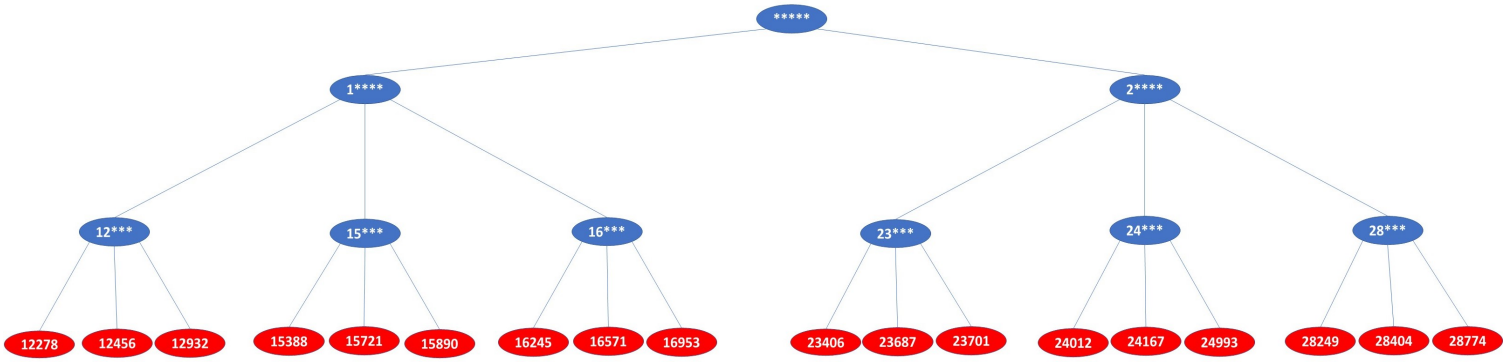

Figure 16: Generalization Hierarchy of zipcode

The following example in figure 19 defines two anonymization models (single-dimensional and multidimensional) for a simple dataset 18.

In single-dimensional anonymization, there are values from specific non-overlapping intervals at each level of the generalization hierarchy, in contrast to multidimensional anonymization, in which overlapping intervals are allowed in the generalization hierarchy.

To define the correct partitions in single-dimensional model, the domain is divided by drawing parallel lines with respect to the axes, and these lines cross the entire space. Alternatively, in the multidimensional model, two sub-domains are defined, drawing a straight line parallel to one axis. Then, these two sub-domains are retrospectively defined other sub-domains by drawing lines with respect to any axis, as long as these lines do not intersect with other sub-domains. Each record can be represented as a point on the space as shown in the figure 19. To solve k-anonymity it is enough to have at least k records in each subspace.

Mondrian's basic concept is the retrospective separation of -dimensional space using a greedy algorithm. The algorithm follows a few steps below:

1. Chooses the dimension according to which the space will be divided.
2. Implements the partition based on the above dimension, from which two sub-domains S1 and S2 arise.
3. For each of the two sub-domains S1 and S2, the procedure is repeated until there is no other allowable point for separation in any dimension.
4. The optimal multidimensional partition is constructed and therefore the appropriate multidimensional generalization to be used.

Following the above procedure, the algorithm manages to find the optimal multidimensional partition, in each region of which more than k records belong, and therefore k-anonymity is satisfied.

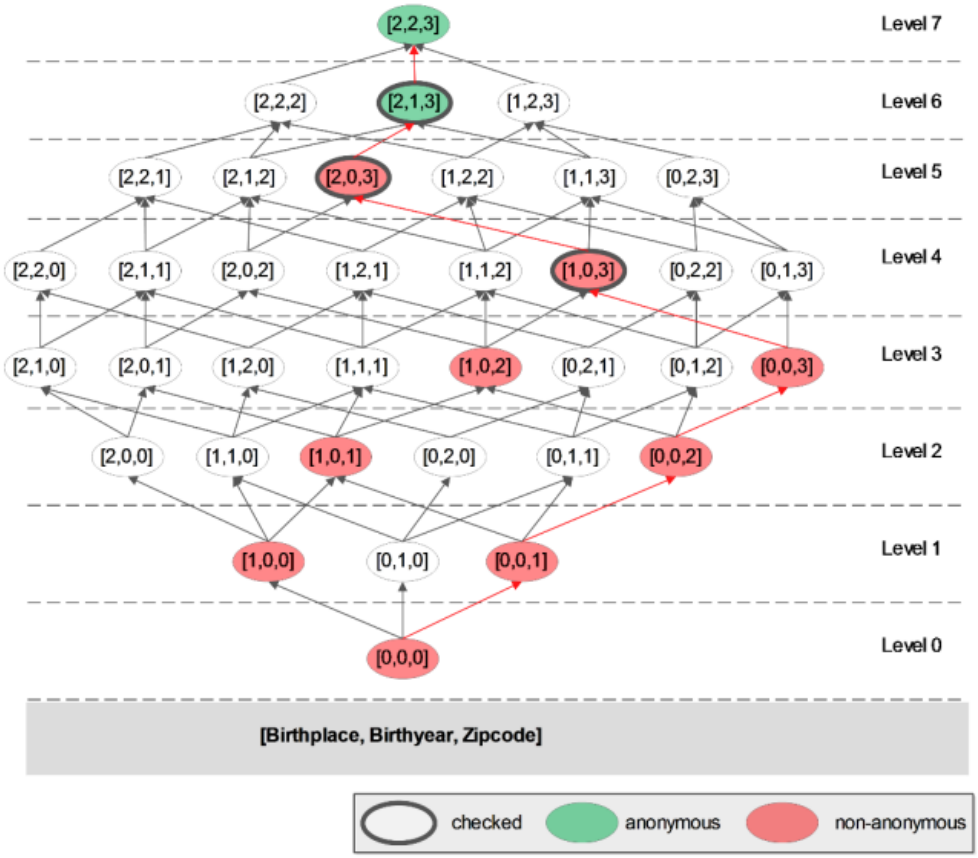

Figure 17: Generalization Lattice with the first iteration of Flash

Although optimal multidimensional partitioning is NP-hard, the algorithm provides a simple and efficient greedy approximation algorithm for several general-purpose quality metrics compared to other models that have been proposed. The overall complexity is $O(n \log n)$ where n is the number of records in the dataset and it is a quite satisfactory time execution.

2.4.4 Datafly Algorithm

Datafly algorithm is an algorithm for providing anonymity in medical data proposed by Latanya Sweeney [26]. Anonymization is achieved by automatically generalizing, substituting, inserting, and removing information as appropriate without losing many of the details found within the data. The method can be used on-the-fly in role-based security within an institution, and in batch mode for exporting data from an institution. To achieve generalization and suppression of records, this algorithm uses a three-step process.

1. A frequency list is created which holds the unique combinations of the quasi identifier set created in the second step of the Datafly-prior process. Each entry in the frequency list corresponds to one or more records in the original dataset.
2. Using domain generalization defined for each quasi-identifiers, the generalization is

Id	Age	Zipcode	Diagnosis
1	21	14851	Heart Disease
2	29	14852	Heart Disease
3	21	14852	Viral Inflection
4	29	14851	Viral Inflection
5	55	14853	Cancer
6	34	14852	Heart Disease
7	34	14853	Viral Inflection
8	55	14851	Viral Inflection

Figure 18: Simple table

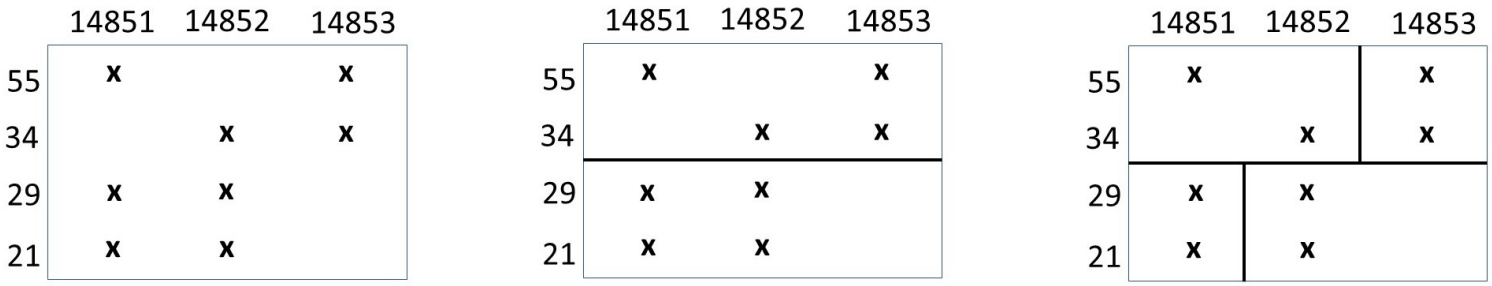

Figure 19: Mondrian example

made. The attribute with the most distinct values is generalized first. This step is run recursively till k or fewer records are having a unique combination of values.

3. All records with unique sequences which have a frequency less than k are suppressed.

The complexity of the Datafly algorithm is $O(nlogn)$. However, this algorithm performs unnecessary generalizations and as so, it does not provide an optimal solution even the solution satisfies k-anonymity. This is one of the biggest problems with Datafly as it may generate a solution with high information loss. To illustrate the whole procedure in image 20 we are using the table 1 with quasi identifiers *zipcode* and *age*, their generalization hierarchies as represented in figures 10 11 and $k = 2$.

2.5 Attacks on K-anonymous Table

K-anonymity offers simple protection and easy to understand it. If a dataset satisfies k-anonymity, then everyone who knows only quasi-identifiers' values of an individual, can not identify the appropriate record for that individual with a certainty greater than $1/k$. While k-anonymity protects identity disclosure, it does not provide adequate protection

Iteration = 1

Age	Post-Code	Frequency
44	12278	1
57	23448	1
53	31284	1
45	45026	1
49	45354	1
42	12456	1
59	23781	1
54	31695	1

Iteration = 2

Age	Post-Code	Frequency
44	12***	1
57	23***	1
53	31***	1
45	45***	1
49	45***	1
42	12***	1
59	23***	1
54	31***	1

Iteration = 3

Age	Post-Code	Frequency
[40-45)	12***	2
[55-60)	23***	2
[50-55)	31***	2
[45-50)	45***	2

Figure 20: Datafly's iterations

against attribute exposure. There are two main major attacks known as Homogeneity and Background Knowledge attacks; let us discuss them based on the 3-anonymous table 2 which contains medical data of a hospital.

2.5.1 Homogeneity Attack

Homogeneity attack showed that when there is little diversity in the sensitive attributes, the adversary can identify the value of the sensitive attribute for that group of k-records. Suppose we have a group of k different files and they all share a specific pseudo-ID. An attacker cannot identify the person based on pseudo-IDs. However, if the main interest of the attacker is the sensitive features and all groups have the same value then, the data has been leaked.

For instance, a politician who intends to be elected to a post in the governance of municipality of Athens utilizes the medical history of his opponent in demonstrating to the populace that his opponent cannot or is not ready to deal with the obligations as an agent of the municipality due to his medical problems. He will have to search for his opponent's medical information by utilizing the released data of the 3-anonymous table from the hospital 2. Despite the likelihood that the data is a 3-anonymized table. Since he has some information about his opponent, he can recognize what ailment his opponent has because when there is no much contrasts (there is little diversity) in the sensitive data. Case in point, he knows that the patient is 37 years old who lives in the postal division 11633 at Athens, so due to this current data, he realizes that his rival has Cancer.

2.5.2 Background Knowledge Attack

In this attack, the adversary uses background knowledge to make the attack successful, and we will show that k-anonymity does not guarantee privacy against background knowledge attacks. To give an example based on table 2, a woman whose director's father is sick needs

Direct-Identifier	Quasi-Identifiers				Sensitive
Full-name	Age	Post-code	Gender	Municipality	Disease
*	$(20-40]$	116^{**}	$*$	Athens	Cancer
*	$(20-40]$	116^{**}	$*$	Athens	Cancer
*	$(20-40]$	116^{**}	$*$	Athens	Cancer
*	$(40-60]$	151^{**}	$*$	Marousi	Covid-19
*	$(40-60]$	151^{**}	$*$	Marousi	Covid-19
*	$(40-60]$	151^{**}	$*$	Marousi	Covid-19
*	$(60-80]$	185^{**}	$*$	Pireus	Anaemia
*	$(60-80]$	185^{**}	$*$	Pireus	Anaemia
*	$(60-80]$	185^{**}	$*$	Pireus	Broken Leg

Table 2: 3-Anonymous Medical Dataset

to comprehend the nature of the sickness. She knows that her manager's dad is old and he lives at Pireus so she can conclude that he is suffering from either anaemia or he broke his leg. Nonetheless, it is realized that her boss has thalassemia which is a type of inherited anemia. Therefore, it is easy for her to conclude that her director's father has anaemia. Using background knowledge, she distinguishes what malady her colleague's dad has. To summarise, from the above examples, it can be seen that k-anonymity does not guarantee privacy preservation.

2.5.3 l-Diversity

Privacy protection's purpose in a set of records is not only the security against identifying records. It is also the assurance that the attacker will not be able to find easily personal information about an individual from this dataset. A new definition in privacy protection that solved the above problems of revealing sensitive traits consistent with k-anonymity is l-diversity. As the name indicates, it ensures the diversity of the sensitive attributes' values in each equivalence class while maintaining the minimum size of the k-set. The main concept of this method is that there should be diversity in the values of sensitive data greater than or equal to l for each subgroup created by k-anonymity.

According to the definition of l-diversity [27], a data table $T = \{A_1 ... A_n, S\}$, where S is a sensitive attribute, is l-diverse if each q*-block equivalence class of quasi-identifier of the anonymous table $T^* = \{A_1 ... A_n, S\}$ is l-diverse, i.e. contains at least l "well-represented" values for the sensitive attribute S.

To make this more comprehensible, let us consider an example of a homogeneity attack on a 4 Anonymous table 3, with yellow color is the problematic equivalence class from which an attacker can easily identify that someone has cancer despite the satisfaction of 4-anonymity.

-	Direct Identifier	Quasi-Identifiers				Sensitive
Row number	Full-name	Age	Post-code	Gender	Municipality	Disease
1	$*$	$[15-30)$	116^{**}	$*$	$*$	Anaemia
2	$*$	$[15-30)$	116^{**}	$*$	$*$	Anaemia
3	$*$	$[15-30)$	116^{**}	$*$	$*$	Covid-19
4	$*$	$[15-30)$	116^{**}	$*$	$*$	Covid-19
5	$*$	$[40-60)$	1858^{*}	$*$	$*$	Cancer
6	$*$	$[40-60)$	1858^{*}	$*$	$*$	Anaemia
7	$*$	$[40-60)$	1858^{*}	$*$	$*$	Covid-19
8	$*$	$[40-60)$	1858^{*}	$*$	$*$	Covid-19
9	$*$	$[30-40)$	116^{**}	$*$	$*$	Cancer
10	$*$	$[30-40)$	116^{**}	$*$	$*$	Cancer
11	$*$	$[30-40)$	116^{**}	$*$	$*$	Cancer
12	$*$	$[30-40)$	116^{**}	$*$	$*$	Cancer

Table 3: 4-Anonymous Medical Dataset-Homogeneity Attack

In order to deal with the above problem, we can choose a different generalization approach and randomize the row order, so the following table 4 is obtained, which satisfies the l-divarsity with $l = 3$. Each equivalence class contains at least three different values for the sensitive attribute *Disease*

2.5.3.1 Distinct l-Diversity

Distinct l-Diversity is indeed the simplest form, with the "well-represented" values being considered the l distinct values. In this case, there is no limit to the occurrence frequency of each sensitive value in every equivalence class. As a result, if a value has a very high frequency of occurrence, it is possible to draw a conclusion from someone with knowledge of the distribution of sensitive values. This led to the development of the following two strongest concepts of l - diversity.

2.5.3.2 Entropy l-Diversity

In order for an anonymized table to be considered $T^* = \{A_1 ... A_n, S\}$ l-diverse with entropy, the relation below must apply to each equivalence class (q*-block) [27].

$$-\sum_{s \in S} p_{(q^*,s)} \cdot \log p_{(q^*,s')} \geq \log(l)$$

where $p_{(q^*,s)} = \frac{n_{(q^*,s)}}{\sum_{s' \in S} n_{(q^*,s')}}$ is the fraction of records in the q*-block with sensitive attribute

-	Direct Identifier	Quasi-Identifiers				Sensitive
Row number	Full-name	Age	Post-code	Gender	Municipality	Disease
10	$*$	$[15-40)$	1163^*	$*$	$*$	Cancer
9	$*$	$[15-40)$	1163^*	$*$	$*$	Cancer
4	$*$	$[15-40)$	1163^*	$*$	$*$	Covid-19
1	$*$	$[15-40)$	1163^*	$*$	$*$	Anaemia
6	$*$	$[40-60)$	1858^*	$*$	$*$	Anaemia
8	$*$	$[40-60)$	1858^*	$*$	$*$	Covid-19
5	$*$	$[40-60)$	1858^*	$*$	$*$	Cancer
7	$*$	$[40-60)$	1858^*	$*$	$*$	Covid-19
3	$*$	$[15-40)$	1169^*	$*$	$*$	Covid-19
11	$*$	$[15-40)$	1169^*	$*$	$*$	Cancer
12	$*$	$[15-40)$	1169^*	$*$	$*$	Cancer
2	$*$	$[15-40)$	1169^*	$*$	$*$	Anaemia

Table 4: 3-Diversity Dataset

value equal to s and $n_{(q^*,s)}$ the number of tuples of T^* from the equivalence class q* with sensitive attribute value s. As a consequence of this condition, every q*-block has at least l distinct values for the sensitive attribute. Thus, the entropy of the whole dataset must be at least log (l). Sometimes this can be very restrictive, as the entropy of the whole table can be small if a few values are very common. So, we are led to the following, the less restrictive notion of l-diversity.

2.5.3.3 Recursive (c, l)-Diversity

An anonymous table $T^* = A_1, ..., A_n, S$ satisfies the recursive (c, l) diversity if for each equivalence class q* the following relation is fulfilled [27]

$$r_1 < c\left(r_\ell + r_{\ell+1} + \cdots + r_m\right)$$

where c is a given constant and r_i is the rate of occurrence of the i^{th} most frequently displayed value for the sensitive attribute S within the equivalence class.

For the recursive (c, l)-Diversity, there are two different ways of application depending on how the data mining is applied to a published table T^*. Publishing an anonymous T^* table leads to a positive disclosure if the attacker can correctly recognize the value of a sensitive feature with high probability. In contrast, a negative disclosure leads when the attacker can confidently exclude some values of a sensitive feature.

2.5.3.4 Positive Disclosure-Recursive (c, l)-Diversity

Therefore in cases where positive disclosure is allowed, the technique of positive disclosure-recursive (c, l)-diversity is applied. Corresponding to the above definitions, and in this concept in order for an anonymized table T^* to satisfy the positive disclosure-recursive (c, l) -diversity, all its equivalence classes must satisfy the positive disclosure-recursive (c, l)-diversity, i.e. one of the following relations applies [27]:

$$y \leq \ell - 1 \text{ and } r_y < c \sum_{j=\ell}^{m} r_j$$

$$y > \ell - 1 \text{ and } r_y < c \sum_{j=\ell-1}^{y-1} r_j + c \sum_{j=y+1}^{m} r_j$$

where Y is a subset of the values of the sensitive trait S for which positive disclosure is allowed, y is the most frequently appearing sensitive value of the equivalence class $q*$, which does not belong to the set Y, r_i is the occurrence frequency of the i^{th} most frequently displayed value within the equivalence class $q*$ and c is a given constant.

2.5.3.5 Negative/Positive Disclosure-Recursive (c_1, c_2, l)-Diversity

Finally, the Negative/Positive Disclosure-Recursive (c_1, c_2, l)-Diversity is defined for a subset W of values of the sensitive feature S for which negative disclosure is not allowed. An anonymized table T^* satisfies the negative/positive disclosure-recursive (c_1, c_2, l)-diversity if it satisfies the positive disclosure-recursive (c, l)-Diversity and each value $s \in W$ is appeared in proportion at least c_2 in the records of each equivalence class.

As an extension of k-anonymity, the main advantage of l-Diversity is the insurance of data-subject's privacy and anonymity, and the avoidance of attacks from drawing conclusions and recognizing sensitive attributes.

2.5.3.6 l-Diversity's weaknesses

However, the technique is insufficient to prevent attribute disclosure, as it remains vulnerable to both skewness and similarity attacks, whereas in many cases, applying l-Diversity can be difficult and unnecessary such as implementing l-Diversity to Big Data [28]. For instance, suppose that in an equivalence class there are 1000 records. In the *Disease* there is a hundred records of Cancer, another hundred for Anaemia and the other eight hundred are Covid-19. This class satisfies the 3-diversity, but the attacker can conclude with 80% certainty that the target person's disease is the Covid-19. Another significant drawback is the high degree of complexity when a data table contains more than one sensitive column.

-	Direct Identifier	Quasi-Identifiers		Sensitive	
Row number	Full-name	Age	Post-code	Monthly Income	Disease
1	*	[15 – 30)	476**	500	Lung Cancer
2	*	[15 – 30)	476**	700	Pneumonia
3	*	[15 – 30)	476**	650	Pulmonary Edema
4	*	[40 – 60)	4790*	1,200	Pneumonia
5	*	[40 – 60)	4790*	1,250	Anaemia
6	*	[40 – 60)	4790*	980	Bronchitis
7	*	[30 – 40)	476**	1,450	Bronchitis
8	*	[30 – 40)	476**	1,000	Anaemia
9	*	[30 – 40)	476**	1,100	Lung Cancer

Table 5: 3-Diverse Dataset-Similarity Attack

2.5.3.7 Skewness attack

When the overall distribution is distorted, l-Diversity will not prevent attributes disclosure. For example, consider a dataset containing data for 1000 patients. The quasi-identifiers in the dataset are Age, Height and Weight. There is a single confidential attribute AIDS whose values can be "Yes" or "No" with 99% being negative, and only 1% positive. Assuming that an equivalence class has the same number of negative and positive records and it satisfies distinct 2-diversity, entropy 2-diversity, and any recursive (c, 2)-diversity can be imposed. However, this is a serious threat to privacy, because everyone in the class can be considered to have a 50% chance of being positive, compared to 1% of the total population.

2.5.3.8 Similarity attack

When the sensitive features' values in the equivalence class are distinct but semantically similar, the attacker is capable to learn important information about an individual. Let examine the following example, with the depicted data table 5 which satisfies 3-Diversity with sensitive attributes *Disease* and *Monthly Income*.

If the attacker knows that Nick is 27 years old and lives in the area with postal code 47678 then he belongs to the first class of equivalence of the table 5. From the above information the attacker is led to the conclusion that Nick's income is relatively low and he suffers from lung-related disease. So, l-diversity can ensure the diversity of sensitive values in each group, but it does not take into account the semantic proximity of these values.

2.5.4 t-Closeness

To prevent the limitations of l-diversity proposed a notion of privacy called t-closeness [29]. As a result of the above l-diversity's problems, distributions that have the same level of diversity offer different levels of privacy, and these are:

- Depending on the semantic relations between the sensitive values.
- The different sensitivity levels of the fields.
- The total data distribution of the dataset.

The basic principle of the technique is that the distribution of a sensitive attribute in each equivalence class of the data table should be as close as possible to the distribution of that attribute in the overall data table. Therefore in the t-closeness, the values of the sensitive features in each equivalence class must not only be l-diverse from each other but must have the same occurrence frequency as the original table.

We must now determine the appropriate metric for the distance between two probabilistic distributions, which will reflect the semantic distance between the sensitive attributes values. Consequently, according to the definition [29], a table T^* satisfies the t-closeness if each of its equivalence classes satisfies the t-closeness, i.e. if the distance between the distribution of the sensitive feature within the equivalence class and its distribution throughout the table is not greater than the limit t. The lower the t value, the closer the two distributions are, which means the more secure anonymization is achieved.

The appropriate metric for calculating the distance between two probabilistic distributions is EMD (Earth Mover's Distance) [30], The EMD is based on the minimal amount of work needed to transform one distribution to another by moving distribution mass between each other and formally it is defined as follows.

Let $P = (p_1, p_2, ...p_m)$, $Q = (q_1, q_2, ...q_m)$, and d_{ij} be the ground distance between element i of P and element j of Q. We want to find a flow $F = [f_{ij}]$ where f_{ij} is the flow of mass from element i of P to element j of Q that minimizes the overall work:

$$EMD[P, Q] = WORK(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} f_{ij}$$

subject to the following constraints:

$$f_{ij} \geq 0 \quad 1 \leq i \leq m, \quad 1 \leq j \leq m$$
$$p_i - \sum_{j=1}^{m} f_{ij} + \sum_{j=1}^{m} f_{ji} = q_i \quad 1 \leq i \leq m$$
$$\sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} = \sum_{i=1}^{m} p_i = \sum_{i=1}^{m} q_i = 1$$

To use t-closeness with EMD, we need to be able to calculate the EMD between two distributions. One can calculate EMD using solutions to the transportation problem, such as a min-cost flow [31]. However, these algorithms do not provide an explicit formula. For this reason, different algorithms for calculating the EMD metric are proposed below, depending on the data type of the sensitive attribute.

2.5.4.1 Calculation of EMD metric for Numerical sensitive attributes

In case that the sensitive feature data are numerical, we will calculate the metric EMD using the ordered distance. So, we consider that numerical attribute values are ordered. Let the attribute domain be $v_1, v_2 ... v_m$, where v_i is the i^{th} smallest value. Ordered distance is defined as the distance between two values equal to the number of values between them in the total order, i.e.

$$\text{ordered\_dist}(v_i, v_j) = \frac{|i-j|}{m-1}$$

To calculate EMD under ordered distance, we only need to consider flows that transport distribution mass between adjacent elements, $P = (p_1, p_2, ..., p_m)$ to $Q = (q_1, q_2 ..., q_m)$ achieving the smallest possible work when they are transferred sequentially, so the distance between the two distributions can be calculated from the following function [29]:

$$\text{EMD}[\mathbf{P}, \mathbf{Q}] = \frac{1}{m-1}\left(|r_1| + |r_1 + r_2| + \ldots + |r_1 + r_2 + \ldots r_{m-1}|\right) = \frac{1}{m-1} \sum_{i=1}^{i=m} \left| \sum_{j=1}^{j=i} r_j \right|$$

where $r_i = p_i - q_i, (i = 1, 2, \ldots, m)$

2.5.4.2 Calculation of EMD metric for Categorical sensitive attributes

For categorical attributes, a total order often does not exist. We consider two distance measures.

Equal Distance: The ground distance between any two value of a categorical attribute is defined to be 1. It is easy to verify that this is a metric. As the distance between any two values is 1, for each point that $pi-qi > 0$, one just needs to move the extra to some other points. Thus we have the following formula [29]:

$$\text{EMD}[\mathbf{P}, \mathbf{Q}] = \frac{1}{2}\sum_{i=1}^{m}|p_i - q_i| = \sum_{p_i \geq q_i}(p_i - q_i) = -\sum_{p_i < q_i}(p_i - q_i)$$

Hierarchical Distance: We consider the distance between two alphanumeric values of an identifier and it is based on the minimum required level of generalization so that both values are generalized to the same value according to the domain generalization hierarchy. Given an hierarchy with height H and two distributions $P = (p_1, p_2, ..., p_m)$ and $Q = (q_1, q_2..., q_m)$ we define the extra function as follows [29]:

$$\text{extra}(N) = \begin{cases} p_i - q_i & \text{if N is a leaf} \\ \sum_{C \in \text{Child}(N)} \text{extra}(C) & \text{otherwise} \end{cases}$$

where Child(N) is the set of all leaf nodes below node N. The extra function has the property that the sum of extra values for nodes at the same level is 0. We further define two other functions for internal nodes [29]:

$$\text{pos\_extra}(N) = \sum_{C \in \text{Child}(N) \wedge \text{extra}(C) > 0} |\text{extra}(C)|$$
$$\text{neg\_extra}(N) = \sum_{C \in \text{Child}(N) \wedge \text{extra}(C) < 0} |\text{extra}(C)|$$

The EMD metric can be calculated as the sum of the movement costs between the descendants of N, i.e. [29]

$$\text{EMD}[\mathbf{P}, \mathbf{Q}] = \sum_{N} \text{cost}(N)$$

where: $\text{cost}(N) = \frac{\text{height (N)}}{H} \min(\text{ pos\_extra } (N),\ \text{neg\_extra } (N))$ and N is a non-leaf node.

Let examine the example in [29] according to the table 7 which is the 3-Diverse version of table 6, there is a semantic problem in the first equivalence class where an attacker is capable to conclude that an individual who belongs in that class has a stomach disease and has low salary income. The distance between the distribution gastric ulcer,gastritis, stomach cancer and the overall distribution is 0.5, while the distance between the distribution gastric ulcer, stomach cancer, pneumonia is 0.278. Respectively, the distance for the distribution 30.000, 40.000, 50.000 is 0.375 whereas for the distribution 60.000, 110.000, 80.000 is 0.278.

	ZIP Code	Age	Salary	Disease
1	47677	29	3 K	gastric ulcer
2	47602	22	4 K	gastritis
3	47678	27	5 K	stomach cancer
4	47905	43	6 K	gastritis
5	47909	52	11 K	flu
6	47906	47	8 K	bronchitis
7	47605	30	7 K	bronchitis
8	47673	36	9 K	pneumonia
9	47607	32	10 K	stomach cancer

Table 6: Original Medical Data

	ZIP Code	Age	Salary	Disease
1	476**	2*	3 K	gastric ulcer
2	476**	2*	4 K	gastritis
3	476**	2*	5 K	stomach cancer
4	4790*	≥ 40	6 K	gastritis
5	4790*	≥ 40	11 K	flu
6	4790*	≥ 40	8 K	bronchitis
7	476**	3*	7 K	bronchitis
8	476**	3*	9 K	pneumonia
9	476**	3*	10 K	stomach cancer

Table 7: 3-Diverse Dataset of Table 6-Similarity Attack

	ZIP Code	Age	Salary	Disease
1	4767*	≤ 40	3 K	gastric ulcer
3	4767*	≤ 40	5 K	stomach cancer
8	4767*	≤ 40	9 K	pneumonia
4	4790*	≥ 40	6 K	gastritis
5	4790*	≥ 40	11 K	flu
6	4790*	≥ 40	8 K	bronchitis
2	4760*	≤ 40	4 K	gastritis
7	4760*	≤ 40	7 K	bronchitis
9	4760*	≤ 40	10 K	stomach cancer

Table 8: Table that has 0.167-closeness for Salary and 0.278-closeness for Disease

In order to minimize the values of t in the first equivalence class, the table is reconstructed as shown in the table 8 .

The Similarity Attack is prevented in Table 8. For instance, an attacker can not infer that an individual has a low salary or has stomach-related diseases. It is noted that t-closeness protects against attribute disclosure, but does not deal with identity disclosure. Thus, it may be desirable to use both t-closeness and k-anonymity at the same time.

2.5.5 Anatomy

In data anonymization, a large part of useful information is lost, as a result of which they can not be utilized. This is caused by data generalizations in order to create equivalence classes. In addition, the correlation of each record with its sensitive value is often not protected.

The basic concept of Anatomy [32] is the data publication without generalization but separated in two tables and linked together by a grouping mechanism. In this way the information loss is reduced because in the published records the original values of quasi-identifiers and sensitive attributes remain the same, however the correlation of every record with its sensitive value is hidden.

More specifically, the procedure is analyzed in a few steps below [32]:

1. Sensitive features are separated from the data table.
2. Two sub-tables are created, the former contains the set of quasi-identifiers in the data sample called quasi-identifier table (QIT), the latter contains the sensitive attribute of the original dataset called sensitive table (ST).
3. The data of QIT is divided into equivalence classes in such a way that each element of the table belongs to a single equivalence class. Moreover, l-diversity is applied on every record and an identifier number (id) is assigned on them.

Quasi-Identifiers			Sensitive
Age	Sex	Zipcode	Disease
23	M	11000	pneumonia
27	M	13000	dyspepsia
35	M	59000	dyspepsia
59	M	12000	pneumonia
61	F	54000	flu
65	F	25000	gastritis
65	F	25000	flu
70	F	30000	bronchitis

Table 9: Original Medical Dataset

Quasi-Identifiers			Sensitive
Age	Sex	Zipcode	Disease
[21, 60]	M	[10001, 60000]	pneumonia
[21, 60]	M	[10001, 60000]	dyspepsia
[21, 60]	M	[10001, 60000]	dyspepsia
[21, 60]	M	[10001, 60000]	pneumonia
[61, 70]	F	[10001, 60000]	flu
[61, 70]	F	[10001, 60000]	gastritis
[61, 70]	F	[10001, 60000]	flu
[61, 70]	F	[10001, 60000]	bronchitis

Table 10: 4-anonymous and 2-diverse table

4. In the QIT table, which includes the accurate values of the quasi-identifiers, a column is added that contains the number that identifies which equivalence class each record belongs to.

5. The ST table contains the sensitive attribute values and two additional columns, the first holds the identifier number (id) of equivalence class to which every sensitive value belongs to, and the other columns depicts the frequency of every value in each equivalence class.

Let's focus on the following example according to [32], the table 9 demonstrates the data of a hospital the quasi-identifiers are *Age*, *Sex*, *Zipcode* and the sensitive attribute is *Disease*

Initially, the table 9 is generalized in such a way as to create equivalence classes that satisfy 4-anonymity and 2-diversity, as shown below in the table 10.

Then, the table 11 (QIT) is created with the original values of the quasi-identifiers , adding a new column containing the number of the equivalence class (id) to which each record belongs.

Age	Sex	Zipcode	Group-ID
23	M	11000	1
27	M	13000	1
35	M	59000	1
59	M	12000	1
61	F	54000	2
65	F	25000	2
65	F	25000	2
70	F	30000	2

Table 11: The quasi-identifier table (QIT)

Group-ID	Disease	Count
1	dyspepsia	2
1	pneumonia	2
2	bronchitis	1
2	flu	2
2	gastritis	1

Table 12: The sensitive table (ST)

In the last step, the sensitive table (ST) 12 is formed, which includes the initial values of the sensitive feature, together with the number of the equivalence class (id), and the number of occurrences of the specific value within the equivalent class.

Using anatomy and publishing the last two tables, the attacker on the one hand, knowing some of the values of the quasi-identifier, can determine if the person he is looking for belongs to a record, on the other hand the attacker can not relate any record with the sensitive value of the equivalence class to which it belongs, since each group satisfies the 2-diversity. Anatomy is preferred over generalization, in cases where the attacker knows quasi-identifiers values of a record and he is certain that the target person is in the published records.

2.6 m-Invariance

The m-invariance [33] technique was created to ensure anonymity in non-static data publications, i.e. in cases when the published data table needs to be updated periodically depending on the nature of the data and how they change at predetermined time periods.

None of the above methodologies support the republishing of data after any changes to the database, such as adding and deleting records. The m-invariance is an extension of the l-diversity [27], so that dynamic data can also be anonymized.

The first attempt to create a technique that would anonymize incremental datasets took

place in 2006 by J-W. Byun, Y.Sohn, E.Bertino and N. Li [34]. However, this concept was not considered enough effective , as it concerns data lists that are constantly growing, i.e. it refers only to the addition of new data without taking into account their modification or deletion. For this reason in 2007 it was proposed the m-invariance by Xiaokui Xiao and Yufei Tao [33].

In the following lines, basic and useful concepts for the definition of m-invariance are presented as formulated in [33].

- **Generalized Historical Union:** Given a generalized relation $T^*(j)(1 \leq j \leq n)$, we convert each row $t^* \in T^*(j)$ to a timestamped tuple $< t^*, j >$, which augments t^* with another attribute A^{tm}, called "Timestamp", storing j. The generalized historical union $U^*(n)$ includes all the timestamped tuples converted from $T^*(1), ..., T^*(n)$, or formally:

$$U^*(n) = \bigcup_{j=1}^{n} \left(\bigcup_{t^* \in T^*(j)} \langle t^*, j \rangle \right)$$

- **Lifespan:** Each tuple $t \in U(n)$ is implicitly associated with a lifespan $[x, y]$, where $x(y)$ is the smallest (largest) integer j such that t appears in $T(j)$.
- **Signature:** Let QI^* be a QI group in $T^*(j)$ for any $j \in [1, n]$. The signature of QI^* is the set of distinct sensitive values in QI^*.

Now, according to the above definitions we can define the m-invariance concept.

A generalized table $T^*(j)(1 \leq j \leq n)$ is **m-unique**, if each QI group in $T^*(j)$ contains at least m tuples, and all the tuples in the group have different sensitive values. A sequence of published relations $T^*(1), ..., T^*(n)$ (where $n \geq 1$) is **m-invariant** if the following conditions hold:

1. $T^*(j)$ is m-unique for all $j \in [1, n]$.
2. For any tuple $t \in U(n)$ with lifespan $[x, y]$, $t.QI^*(x), t.QI^*(x+1), ..., t.QI^*(y)$ have the same signature, where $t.QI^*(j)$ is the generalized hosting group of t at time $j \in [x, y]$.

The difficulty of applying the m-invariance technique lies in fulfilling the second condition of the above definition, as two equivalence classes can not have the same signature when there are records deletions that may remove a single sensitive value from the updated data list which will be published.

These cases, called "critical absence" and leading to a possible exposure of personal information, are treated by adding counterfeit tuples to the published table when applying m-invariance, which have the deleted sensitive values. Along with each published table containing counterfeit tuples, another table is released containing the number of counterfeit tuples for each equivalence class.

Name	Age	Zip.	Disease
Bob	21	12000	dyspepsia
Alice	22	14000	bronchitis
Andy	24	18000	flu
David	23	25000	gastritis
Gary	41	20000	flu
Helen	36	27000	gastritis
Jane	37	33000	dyspepsia
Ken	40	35000	flu
Linda	43	26000	gastritis
Paul	52	33000	dyspepsia
Steve	56	34000	gastritis

G.ID	Age	Zip.	Disease
1	[21, 22]	[12k, 14k]	dyspepsia
1	[21, 22]	[12k, 14k]	bronchitis
2	[23, 24]	[18k, 25k]	flu
2	[23, 24]	[18k, 25k]	gastritis
3	[36, 41]	[20k, 27k]	flu
3	[36, 41]	[20k, 27k]	gastritis
4	[37, 43]	[26k, 35k]	dyspepsia
4	[37, 43]	[26k, 35k]	flu
4	[37, 43]	[26k, 35k]	gastritis
5	[52, 56]	[33k, 34k]	dyspepsia
5	[52, 56]	[33k, 34k]	gastritis

Table 13: Initial table $T(1)$ and Generalized table $T^*(1)$ at the 1st release

Name	Age	Zip.	Disease
Bob	21	12000	dyspepsia
David	23	25000	gastritis
Emily	25	21000	flu
Jane	37	33000	dyspepsia
Linda	43	26000	gastritis
Gary	41	20000	flu
Mary	46	30000	gastritis
Ray	54	31000	dyspepsia
Steve	56	34000	gastritis
Tom	60	44000	gastritis
Vince	65	36000	flu

G.ID	Age	Zip.	Disease
1	[21, 23]	[12k, 25k]	dyspepsia
1	[21, 23]	[12k, 25k]	gastritis
2	[25, 43]	[21k, 33k]	flu
2	[25, 43]	[21k, 33k]	dyspepsia
2	[25, 43]	[21k, 33k]	gastritis
3	[41, 46]	[20k, 30k]	flu
3	[41, 46]	[20k, 30k]	gastritis
4	[54, 56]	[31k, 34k]	dyspepsia
4	[54, 56]	[31k, 34k]	gastritis
5	[60, 65]	[36k, 44k]	gastritis
5	[60, 65]	[36k, 44k]	flu

Table 14: Initial table $T(2)$ and Generalized table $T^*(2)$ at the 2nd release

To cite an example, suppose a hospital releases patients data every six months. In Table 13, $T(1)$ is the original table which is used as a basis for anonymizing and publishing Table $T^*(1)$. After six months based on data from the initial table $T(2)$, the $T^*(2)$ is released. The last two datasets are depicted in table 14.

The patients Alice, Andy, Helen, Ken and Paul have been deleted from the database. Respectively, patients Emily, Mary, Ray, Tom and Vince have been added.Even though both published relations (Tables $T^*(1)$, $T^*(2)$) are 2-anonymous and 2-diverse, an adversary can still precisely determine the disease of a patient, by exploiting the correlation between the two "snapshots".

To illustrate this, assume, again, an adversary who has Bob's age and Zip-code, and knows that Bob has a record in both Tables $T^*(1)$ and $T^*(2)$ (i.e., Bob was admitted for treatment,

Name	G.ID	Age	Zip.	Disease
Bob	1	[21, 22]	[12k, 14k]	dyspepsia
c_1	1	[21, 22]	[12k, 14k]	bronchitis
David	2	[23, 25]	[21k, 25k]	gastritis
Emily	2	[23, 25]	[21k, 25k]	flu
Jane	3	[37, 43]	[26k, 33k]	dyspepsia
c_2	3	[37, 43]	[26k, 33k]	flu
Linda	3	[37, 43]	[26k, 33k]	gastritis
Gary	4	[41, 46]	[20k, 30k]	flu
Mary	4	[41, 46]	[20k, 30k]	gastritis
Ray	5	[54, 56]	[31k, 34k]	dyspepsia
Steve	5	[54, 56]	[31k, 34k]	gastritis
Tom	6	[60, 65]	[36k, 44k]	gastritis
Vince	6	[60, 65]	[36k, 44k]	flu

Group − ID	Count
1	1
3	1

Table 15: $T^*(3)$ which is $T^*(2)$ with counterfeits and published counterfeit statistics table

within 6 months before both publication times). Based on Table $T^*(1)$, the adversary is certain that Bob must have contracted either dyspepsia or bronchitis. From Table $T^*(2)$, he/she finds out that Bob's disease must be either dyspepsia or gastritis. By combining the above knowledge, the adversary correctly captures Bob's real disease dyspepsia.

According to the m-invariance method, table $T^*(2)$ is replaced by table $T^*(3)$, as shown above 15. More specifically, table $T^*(3)$ involves a generalized tuple for every row in table $T^*(2)$, together with two counterfeit tuples c_1 and c_2 (names are not published; they are included for row referencing). The 13 tuples are partitioned into six QI groups.

The two releases (tables $T^*(1)$ and $T^*(3)$) have an important property. If a tuple appears in the microdata at both publication timestamps, it is generalized to two QI groups (one per timestamp) containing the same sensitive values. For instance, the tuple <Jane, 37, 33k, dyspepsia> belongs to both Tables $T(1)$ and $T(2)$. It is generalized to QI groups 4 and 3 in Tables $T^*(1)$ and $T^*(3)$, respectively. The two groups include an equivalent set of diseases: {*dyspepsia*, *flu*, *gastritis*} (as is achieved via a counterfeit c_2). As a result, even if an adversary finds out both QI groups, he/she can only conjecture that Jane's disease may be an element in that equivalent set.

To be precise, m-invariance requires the satisfaction of m-diversity and at the same time a record always belong to the same equivalence class, which has the same set of sensitive properties, for all publications.

In this way we avoid personal information leakage but we significantly corrupt the original data, which is the main drawback of this technique. The information loss from the constant generalizations of quasi-identifiers values as well as the gradual accumulation of counterfeits records as the number of publications increases, often leads to information that is impossible to use for any kind of research or study.

2.7 δ-Presence

Another metric used to protect privacy in databases is the δ-presence. The method guarantees that by anonymizing the database, an attacker will not be able to determine if an individual is included in that database with a certainty greater than δ [35] [36]. So, given an external public table P, and a private table T , we say that δ-presence holds for a generalization T^* of T , with $\delta = (\delta_{min}, \delta_{max})$ if

$$\delta_{min} \leq P\left(t \in T \mid T^*\right) \leq \delta_{max} \quad \forall t \in P$$

In such a dataset, we say that each tuple $t \epsilon P$ is δ-present in T. Therefore, $\delta = (\delta_{min}, \delta_{max})$ is a range of acceptable probabilities for $P\left(t \in T \mid T^*\right)$.

The k-anonymity and l-diversity methods guarantee privacy protection under the condition that an attacker knows information about a person and is sure that individual's information is included in the published dataset. However, this is not enough in many cases. Suppose a dataset which contains the cancer patients of a country. The attacker does not have to be sure that the victim is included in this table, because then he/she knows with certainty that the individual has cancer. Other examples of such datasets may be a database containing information about illegal organizations or a set of patients data with a specific type of diabetes. In both cases, specifying that an individual or group is included in the database may be harmful to the individual's privacy.

In [35] examines whether the data is considered sufficiently anonymous, through the analysis of the risk of the participation or not of a natural person in the anonymized data. It is defined as the problem of hiding individuals' presence in a database and proves the impossibility of k-anonymity in publication cases of sensitive attribute values.

2.8 k^m-Anonymity

Despite the various techniques that have been developed, and the various guarantees offered in the literature, many privacy risks remain unaddressed. The available information that the attacker may possess can take many forms. At the same time, the published data models may differ from time to time, with the result that each case requires a different processing in order to ensure the databases privacy.

K^m-anonymity [15] comes to solve such a problem as described above. In this problem each record consists of datasets that get values from a common domain depicted in teble 16. The attacker who knows up to *m* items of a target record, and tries to identify the remaining values of the record and associate the published record with a real person. So, k^m-anonymity provides protection against identity disclosure.

In contrast to previous privacy guarantees, there is no clear distinction between sensitive and quasi-identities. In each case a subset of the records' values forms the set of the quasi-

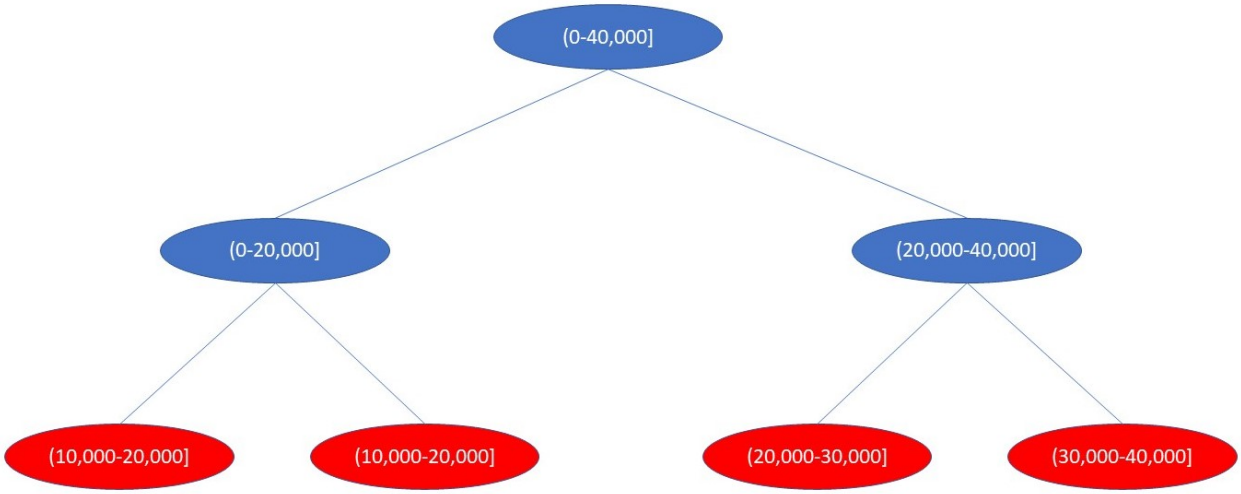

Figure 21: Data Generalization Hierarchy

Name	Payments
John	{11000, 11000, 20000, 40000, 40000}
Mary	{11000, 30500, 40000}
Nick	{11000, 11000, 40000, 40000}
Sandy	{11000}
Mark	{20000}

Table 16: Set-valued dataset-Payment data

identifiers and the remaining values form the set of the sensitive attributes. Each record has a different size, as opposed to relational databases where the size of each record is fixed.

This is a new version of k-anonymity, in which each combination of values with *m* size , appears at least k times in the dataset. K^m-anonymity ensures that any attacker who knows up to *m* items of a target record cannot use that knowledge to identify more than k individuals in the dataset. This guarantee is a relaxation of the classic k-anonymity [1,2]. Consider the 2^2-anonymous Table 17 which is an anonymization of Table 16. An attacker with partial knowledge of up to 2 values of a target, will not be able to identify less than 2 records. To achieve this level of privacy in our dataset, using the data hierarchy of Figure 21, all values had to be generalized because values 20,000 and 30,500 were rare. However, the same privacy can be ensured in Table 18 where values 20,000 and 30,500 are generalized to the range [20,000-30,500] [37]. As we can observe, less values are generalized and a smaller information loss is achieved.

Therefore, according to definition in [15] a dataset D satisfies k^m-anonymity, if every combination of m values is presented in at least k different records.

Id	Payments
1	(10000 – 20000], (10000 – 20000], (30000 – 40000], (30000 – 40000], (30000 – 40000]
2	(10000 – 20000], (30000 – 40000], (30000 – 40000]
3	(10000 – 20000], (10000 – 20000], (30000 – 40000], (30000 – 40000]
4	(10000 – 20000]
5	(10000 – 20000]

Table 17: 2^2-anonymous table using a data generalization hierarchy

Id	Payments
1	11000, 11000, [20000 – 30500], 40000, 40000
2	11000, [20000 – 30500], 40000
3	11000, 11000, 40000, 40000
4	11000
5	[20000 – 30500]

Table 18: 2^2-anonymous table using a dynamic hierarchy

2.8.1 Generalization model

For the anonymization process, the global recording [2,19] technique is selected, according to which a value in the database is replaced by a more general value containing the original, without changing its semantics. The set of possible generalizations of a database is the tree of the generalization hierarchy as shown in the figure 22. The higher the generalization level, the greater the information loss presented by the data. In a data collection, all the values in the database must be in the hierarchy tree as shown in the figure 22.

In this example the dataset D in table 19 does not satisfy the k^m-anonymity, since for $k = 2$ and $m = 2$ the combination of values a1, b1 appears only once. The application of generalization $\{a_1, a_2\} \rightarrow A$ to the database can solve the problem, since now any combination of $m = 2$ values in the database, appears in at least $k = 2$ records as demonstrated in figure 20 .

id	contents
t_1	$\{a_1, b_1, b_2\}$
t_2	$\{a_2, b_1\}$
t_3	$\{a_2, b_1, b_2\}$
t_4	$\{a_1, a_2, b_2\}$

Table 19: Dataset D

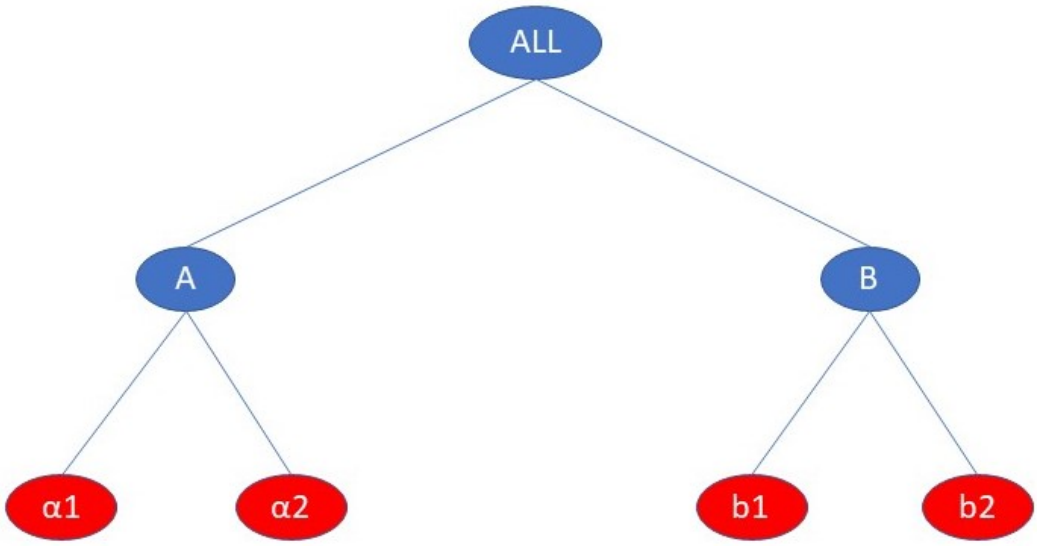

Figure 22: Sample generalization hierarchy

id	contents
t'_1	$\{A, b_1, b_2\}$
t'_2	$\{A, b_1\}$
t'_3	$\{A, b_1, b_2\}$
t'_4	$\{A, b_2\}$

Table 20: Anonymized dataset D

2.8.2 Apriori Algorithm

The Apriori algorithm transforms set-valued data to k^m-anonymous datasets, and exploits the principle of apriori property. According to this property if an itemset J of size i causes a privacy breach, then each superset of J causes a privacy breach. First the algorithm examines the privacy breaches that might be feasible if the adversary knows only 1 item from each trajectory, then 2 and so forth till we examine privacy threats from an adversary that knows *m* items. The benefit of this algorithm is that we can exploit the generalizations performed in step i, to reduce the search space at step i+1.

Unlike the case of previous algorithms such as Flash, Apriori does not perform full-domain hierarchy, but partial-domain hierarchy. The aim of the algorithm is to find a "cut" is the generalization hierarchy, i.e., a horizontal partitioning of nodes as demonstrated in figure 24, that guarantees that if every value is generalized to the most abstract descendant which is under the cut-off line, then the resulting dataset will be anonymous. The solution space is exponential to the size of the domain and the record, thus exhaustive algorithms are very inefficient for realistic datasets. Apriori offers a greedy heuristic with performs efficiently and finds solutions of high quality. It relies on creating a count tree 23, which is a trie tree

that traces the supports of sets of values in the dataset.

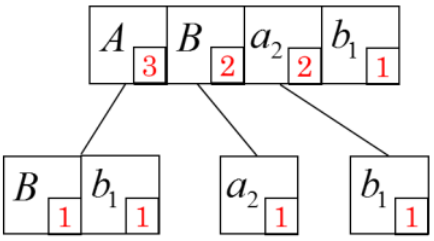

Figure 23: Count-tree

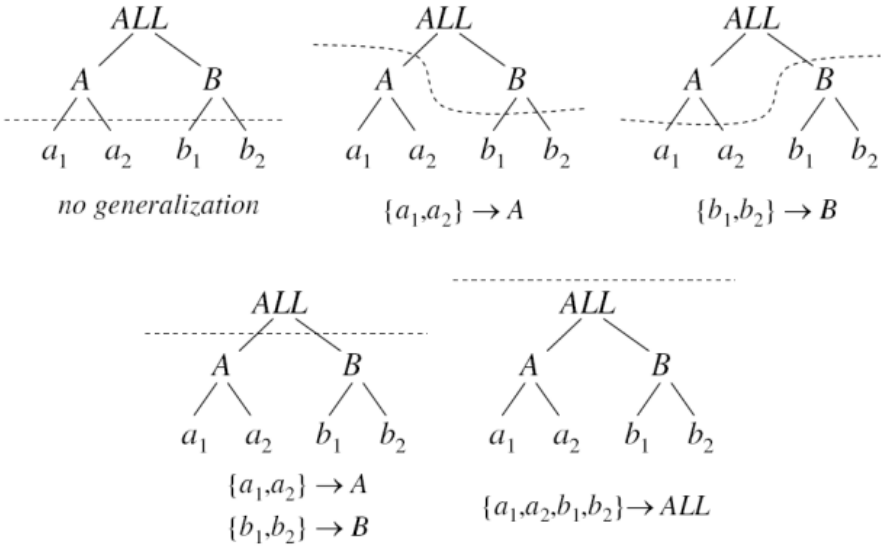

Figure 24: Generalization cuts

More specifically, the algorithm practically iterates the direct algorithm for combination of sizes $i = \{1, ..., m\}$. The database is scanned at each iteration i and the count-tree is populated with itemsets of length i. Then, the algorithm finds in the count-tree the values in the nodes-leaves that have support less than k. Each of these values it refers to the generalization hierarchy tree, and replaces the problematic values with the more generalized ones in order to increase the support of each value to a number greater than k. The algorithm repeats the procedure for all the problematic values of the count-tree. In other word, the basic idea is to iteratively count the supports of sets of increasing length (up to size *m*) and at each step to perform all the necessary generalizations to keep the dataset anonymous. The basic steps of the algorithm are the following in the pseudocode 1 according to [15].

2.9 Randomization

Randomization involves changing the attributes in a dataset to be less precise, while maintaining the overall distribution. More specifically, randomization is a category of anonymization techniques that modify data accuracy to remove the strong connection between the data and the natural person they refer to. Moreover, randomization protects the dataset

Algorithm 1 Apriori-based Anonymization

```
1: procedure AA(D, I, k, m)
2:     initialise c_out                          ▷ c_out: set of generalization rules
3:     for i := 1 to m do
4:         initialize a new count-tree
5:         for all t ∈ D do
6:             extend t according to c_out
7:             add all i-subsets of extended t to count-tree
8:             for all leaves v in count-tree do
9:                 if support(v) < k then
10:                    J := itemset corresponding to v
11:                    find generalization of items in J that make J k-anonymous
12:                    merge generalization rules with c_out
13:                    backtrack to longest prefix of path J, wherein no item has been generalized in c_out
```

from the risk of inference. Examples of randomization techniques include noise addition, permutation and differential privacy. A key advantage of randomization methods is their relatively simple application as will be presented in detail below.

2.9.1 Adding noise

Adding noise is the most studied anonymization technique as many different models have been designed for privacy. The technique of adding noise was initially created to be used in statistical databases with numerical data [38]. However, over the years, after various studies and analyzes, a substantial number of models were created which can be used for categorical data [39].

Before developing the methodology of this technique we will try to clarify the meaning of noise addition. So, adding noise concerns the modifying attributes values in a dataset through the addition of synthetic records with similar quasi-identifiers values. Therefore, the main principle of noise addition is the modification of the attributes of the dataset to make them less accurate.

In the case of databases with numerical data, the addition of noise involves the modifying the values of the quasi-identifiers by a very small number called noise. So the basic logic of the statistical noise addition technique is to integrate n numbers $r_1, r_2, ..., r_n$ from a known distribution into the initial values v_i of a feature $V = \{v_1, v_2, ..., v_n\}$ creating a set of anonymized values of the same feature $U = \{u_1, u_2, ..., u_n\}$. The integration of noise in the initial values of a feature is fulfilled either additively or multiplicatively [40].

In the case of additive noise, the following applies:

$$u_i = v_i + r_i, r \in [-a, a], a \in \mathcal{R}$$

that is, the noise variable belongs to a distribution with an average value of 0 and a very small standard deviation.

In the occasion of multiplicative noise, the following applies:

$$u_i = v_i r_i$$

Where the noise variable belongs to a distribution to which the following applies to the mean and variance:

$$E(u_i \mid v_i) = v_i$$
$$\sigma^2_{u_i|v_i} = V(u_i \mid v_i) = v^2\sigma_r^2$$

As for the alphanumeric data as is evident from the above, the addition of noise alters the data. Therefore, to reduce the effects of data alteration, the statistical data of the sample should be preserved during the application of the technique, ie the average, the variation, etc.

The main benefit of noise addition is that a third party will not be able to identify an individual nor will they be able to restore the data or otherwise discern how the data has been altered if the noise addition is applied effectively and retains the global distribution of the dataset. However, the noise introduced alters the quality of the data as described above, so the analyses performed on the dataset are less relevant. The level of noise depends on the level of information required and the impact that the disclosure of attributes would have on the privacy of individuals.

The basic common mistakes in noise addition can be categorized as follows:

- **Inconsistent noise addition:** If the noise is not semantically viable (i.e. it is disproportionate and does not respect the logic between attributes in a set) or if the data set is too sparse.
- **Assuming that adding noise is sufficient:** Adding noise is a complementary measure that makes it more difficult for an attacker to recover the data, it should not be assumed to be a self-sufficient anonymisation solution.

2.9.2 Permutation

The permutation technique is quite similar to the anatomy technique which described in above section, both maintain the accuracy of the data values.

The key idea of permutation consists of mixing attribute values in a table in such a way that some of them are artificially linked to different data subjects. Permutation, therefore, alters the values within the dataset by simply swapping them from one record to another. More specifically, permutation eliminates the relations between quasi-identifiers and sensitive features by grouping the records into equivalence classes and ”shuffling” the sensitive features of each class [41].

Generally, a permutation on a set V, is an one-to-one mapping from V to itself. If $|V| = N$, there are overall $N!$ permutations on V. Let α be a permutation on a set V of tuples, we use $\alpha(t_i)$ to denote the image of t_i under α. We use S_V to denote the set of all permutations on V. If V is a QI-group of microdata T, for example, $V = QI_i$, we can independently uniformly select a permutation from S_{QI_i} at random [42].

2.9.2.1 Permutation Anonymization

Let T be a table consisting of QI-attributes $A_i (1 \leq i \leq d)$ and sensitive attribute A_s. Given a partition P with m QI-groups on T, permutation anonymization is a procedure with (T, P) as input, which produces a quasi-identifier table PQT and a sensitive table PST satisfying following conditions:

1. PQT is a table with schema $(A_1, A_2, \cdots, A_d$, Group-ID $)$ such that for each tuple $t \in QI_i$, PQT has a tuple of the form: $(\alpha_{i_1}(t) \cdot A_1, \alpha_{i_2}(t) \cdot A_2, \cdots, \alpha_{i_d}(t) \cdot A_d, i)$, where $\{\alpha_{i_j} : (1 \leq j \leq d)\}$ is independently uniformly selected from S_{QI_i} at random.
2. PST is a table with schema (Group-ID, A_s) such that for each tuple $t \in QI_i$, PST has a record of the form: $(i, \alpha_{i_s}(t) \cdot A_s)$, where α_{i_s} is uniformly selected from S_{QI_i} at random.

In simple terms, the table T is split into sensitive attributes and simple quasi-identifiers, creating two subsets of data. The relation between quasi-identifiers and sensitive attributes is removed and two sub-tables are created, one QT, which contains all the quasi-identifiers and the other ST, which contains the sensitive table attribute. Given partition P of the data table T with m quasi-identifiers, the permutation anonymization creates two subsets of data PQT and PST which satisfy the above conditions.

2.9.3 The concept of Differential Privacy

Although k-anonymity is quite a powerful model of anonymization, it has been shown that k-anonymity alone does not always ensure privacy. k-Anonymity is able to prevent identity disclosure, i.e. a record in the k-anonymized dataset cannot be mapped back to the corresponding record in the original dataset. However, in general, it may fail to protect against attribute disclosure [43]. On the contrary Differential Privacy satisfy the demands of being a formal privacy definition.

Differential privacy [44, 45] is a mathematical definition of what it means to have privacy. It is not a specific process like de-identification, but a property that a process can have. For example, it is possible to prove that a specific algorithm "satisfies" differential privacy. Introduced in 2006 [8, 44], Differential Privacy describes a promise to protect individuals from any additional harm that they might face due to their data being in the private database *D* that they would not have faced if their data were not been part of *D*.

Furthermore, differential privacy addresses the paradox of learning nothing about an individual while learning useful information about a population [46]. Informally, differential privacy guarantees the following for each individual who contributes data for analysis: the output of a differentially private analysis will be roughly the same, whether or not you contribute your data. In other words, differential privacy is a powerful, mathematical definition of privacy in the context of statistics and machine learning analysis.

As mentioned above, differential privacy is a powerful standard for data privacy [44, 47], it guarantees that the probability of an attacker who brings good or harm to any set of participants is essentially the same, regardless of whether an individual is inside or outside from the dataset. To achieve this, differential privacy adds a random noise to the query output, so that the difference in output results is covered by the presence or absence of a single person.

Differential privacy has been studied in theory and shown that provides a strict guarantee of privacy, even when the attacker has the background knowledge of the worst-case scenario [8]. It neutralizes all data-linkage attacks and statistical attacks because it uses the property of a data access mechanism that does not depend on the presence or absence of basic background knowledge.

To put it simply, consider an individual who is deciding whether to allow their data to be included in a database. For example, a patient may decide whether their medical records can be used in a study, or someone deciding whether to answer a survey. A useful notion of privacy would be an assurance that allowing their data to be included should have negligible impact on them in the future. As we've already seen, absolute privacy is inherently impossible but what is being guaranteed here is that that the chance of a privacy violation is small. This is precisely what differential privacy provides.

2.9.3.1 Fundamental Properties and Definitions for Differential Privacy

Before defining differential privacy, we need to mention some useful concepts.

It is useful to think of the dataset D as a finite collection of records from a universe U. Let $D = (r_1, r_2, ..., r_n) \in U$ be a dataset in which r_i represents a record or an individual. Informally, any particular row is indiscreet to a differentially private output. Therefore, when viewed from the perspective of any particular data, the calculation is like from a dataset that does not include it. So, we come to the concept of adjacent datasets. Moreover, it is convenient to represent a dataset or database D by its histogram which is a function $h(D) : \mathcal{U} \rightarrow \mathbb{N}^{|\mathcal{U}|}$ in which each entry h_i represents the number of records in the dataset D of type $i \in U$

In order to explain the distance between datasets, we have to define the l_1 norm which a basic term for neighbour datasets.

Definition 2..1 *(l_1 norm) The l_1 norm of a dataset D is denoted $\|D\|_1$ and is defined to be :*

$$\|D\|_1 = \sum_{i=1}^{|D|} |D_i|$$

Definition 2..2 *(Distance Between Databases [46]) The distance between two adjacent or neigh-*

boring datasets D, D' with the l_1 norm is:

$$\|D - D'\|_1 = \sum_{i=1}^{|\mathcal{D}|} |D_i - D'_i| \leq 1$$

Note that $\|D\|_1$ is a measure of the size of a database D (i.e., the number of records it contains), and $|D_i - D'_i|$ is a measure of how many records differ between D and D'.

Before we get into the definition of differential privacy, we will therefore need to discuss the necessity of randomization. More precisely, as we analyzed above any non-trivial privacy guarantee that holds regardless of all present or even future sources of auxiliary information requires randomization. For instance, a non-trivial deterministic algorithm yields different outputs between two dataset under a specific query. Changing one row at a time we see there exists a pair of databases differing only in the value of a single row, on which the same query yields different outputs. An attacker knowing that the database is one of these two almost identical databases learns the value of the data in the unknown row [46]. Thus, it is important to define the input and output space of randomized algorithms. A randomized algorithm with domain A and (discrete) range B will be associated with a mapping from A to the probability simplex over B.

Definition 2..3 *(Probability Simplex [46]) Given a discrete set B, the probability simplex over B, denoted $\Delta(B)$ is defined to be:*

$$\Delta(B) = \left\{ x \in \mathbb{R}^{|B|} : x_i \geq 0 \text{ for all } i \text{ and } \sum_{i=1}^{|B|} x_i = 1 \right\}$$

Definition 2..4 *(Randomized Algorithm [46]) A randomized algorithm M with domain A and discrete range B is associated with a mapping $M : A \to \Delta(B)$. On input $a \in A$, the algorithm M outputs $M(a) = b$ with probability $(M(a))_b$ for each $b \in B$. The probability space is over the coin flips of the algorithm M.*

Hence, the important concept of the above definition where the probability space is over the coin flips of the algorithm M implies that it is the source of randomness.

We are now ready to formally define differential privacy, which intuitively will guarantee that a randomized algorithm produces similar outputs on adjacent inputs. Correspondingly, the influence of any single record-individual on the output of the algorithm is influence and the leakage of any individual's information is prevented.

Definition 2..5 *(Differential Privacy [44, 46]) A randomized algorithm M with domain $\mathbb{N}^{|\mathcal{U}|}$ is (ϵ, δ)-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all adjacent input datasets $D, D' \in \mathbb{N}^{|\mathcal{U}|}$ such that $|D - D'| \leq 1$:*

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta$$

where the probability space is over the coin flips of the mechanism M. *If* $\delta = 0$, *we say that M is* ϵ*-differentially private.*

The parameter ϵ, called the privacy budget and controls how much noise is needed to protect privacy, so it directly impact on accuracy. The noise protects the individual of a data point in the dataset. In general, smaller values of ϵ imply more privacy, as the distributions of algorithm's outputs for neighbouring inputs tend closer. However, the optimal selection of ϵ is open question but there several methods for choosing ϵ [48].

Regarding the parameter δ we are interested in values less than the inverse of any polynomial in the size of the database. In particular, values of δ on the order of $1/\|D\|_1$ are very dangerous, they publish complete records of a small number of database participants. Even when δ is negligible, however, there are theoretical distinctions between $(\epsilon, 0)$- and (ϵ, δ)-differential privacy. Chief among these is what amounts to a switch of quantification order. $(\epsilon, 0)$-differential privacy ensures that, for every run of the mechanism M(D), the output observed is almost equally likely to be observed on every neighboring database, simultaneously. In contrast, (ϵ, δ)-differential privacy says that for every pair of neighboring databases D, D', it is extremely unlikely that, the observed value M(D) will be much more or much less likely to be generated when the database is D than when the database is D'. In other words, (ϵ, δ)-differential privacy allows a privacy leakage with some small probability $delta$.

Now, we will examine some basic properties of differential privacy which make the privacy guarantee resilient to various changes in the algorithm itself, in the group of people to whom the privacy is offered, and when combining the results of a series of algorithms executed on private data. These properties make differential privacy stand out from other anonymization techniques.

- **Post-processing:** Differentially private mechanisms are immune to post-processing. The composition of any function with a differentially private mechanism will remain differentially private. Formally, the composition of a data-independent mapping f with an (ϵ, δ)-differentially private algorithm M is also (ϵ, δ)-differentially private as defined in the lemma which proof can be found in [46].

  Lemma 1 *(Post-processing) Let* $\mathcal{M} : \mathbb{N}^{|\mathcal{U}|} \to Z$ *be a randomized algorithm that is* (ϵ, δ)*-differentially private. Let* $f : Z \to R$ *be an arbitrary randomized mapping. Then* $f \circ \mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R$ *is* (ϵ, δ)*-differentially private.*

  Essentially, what this says is that we may obtain some intermediate private result (in a potentially easy-to-implement way) and then release this result, allowing anyone to perform any further (statistical or other) calculations on their own, without having to worry whether they might process this result in a certain way so as to violate the (ϵ, δ) privacy guarantee

- **Composition:** Perhaps most crucially, the quantification of loss also permits the analysis and control of cumulative privacy loss over multiple computations. Understanding the behavior of differentially private mechanisms under composition enables the design and analysis of complex differentially private algorithms from simpler differentially private building blocks. The composition property splits into sequential composition and parallel composition.

  - **Sequential Composition:** The sequential composition states that if we combine multiple differentially private mechanisms at the same input and release all the results, the privacy budget will add up. Formally, the sequential composition theorem for differential privacy says that [46, 49]:

    Theorem 1 *(Sequential Composition) Let* $\mathcal{M}_i : \mathbb{N}^{|\mathcal{U}|} \to Z_i$, *be an* $(\epsilon_i, 0)$*-differential private algorithm for* $i \in [k]$. *Then their combination defined to be* $\mathcal{M}_{[k]} : \mathbb{N}^{|\mathcal{U}|} \to \prod_{i=1}^{k} \mathcal{Z}_i$ *by the mapping:* $\mathcal{M}_{[k]}(D) = (\mathcal{M}_1(D), \mathcal{M}_2(D), ..., \mathcal{M}_k(D))$ *is* $\left(\sum_{i=1}^{k} \varepsilon_i, 0\right)$ *differentially private.*

    Sequential composition is a vital property of differential privacy because it enables the design of algorithms that consult the data more than once. Sequential composition is also important when multiple separate analyses are performed on a single dataset, since it allows individuals to bound the total privacy cost they incur by participating in all of these analyses. The bound on privacy cost given by sequential composition is an upper bound - the actual privacy cost of two particular differentially private releases may be smaller than this, but never larger.

  - **Parallel Composition:** Parallel composition can be seen as an alternative to sequential composition - a second way to calculate a bound on the total privacy cost of multiple data releases. Parallel composition is based on the idea of splitting your dataset into disjoint chunks and running a differentially private mechanism on each chunk separately. Since the chunks are disjoint, each individual's data appears in exactly one chunk - so even if there are k chunks in total (and therefore k runs of the mechanism), the mechanism runs exactly once on the data of each individual. More formally, parallel composition theorem [46, 50] states that:

    Theorem 2 *(Parallel Composition) Let* $\mathcal{M}_i : \mathbb{N}^{|\mathcal{U}|} \to Z_i$, *be an* $(\epsilon_i, 0)$*-differential private algorithm for* $i \in [k]$. *Let* $D_1, D_2, ..., D_k$ *be* k *partitions of the dataset* D, *such that* $\bigcup_{i=1}^{k} D_i = D$ *and* $\bigcap_{i=1}^{k} D_i = \emptyset$.
    Then their combination $\mathcal{M}_{[k]}(D) = (\mathcal{M}_1(D), \mathcal{M}_2(D), ..., \mathcal{M}_k(D)) : \mathbb{N}^{|\mathcal{X}|} \to \prod_{i=1}^{k} Z_i$ *satisfies* $(\max\{\varepsilon_1, \varepsilon_2, ..., \varepsilon_k\})$*-differential privacy.*

    Note that this is a much better bound than sequential composition would give. Since we run $\mathcal{M}$ k times, sequential composition would say that this procedure satisfies $k\epsilon$-differential privacy. Parallel composition allows us to say that the total privacy cost is just ϵ

- **Group privacy:** Differential privacy permits the analysis and control of privacy loss incurred by groups, such as families. In this case, we can guarantee a similar fact for

groups of k individuals, which is extremely useful. It also addresses the case that multiple k records in the dataset refer to the same individual [46].

Theorem 3 *Any (ϵ,0)-differentially private mechanism is ($k\epsilon$,0)-differentially private for groups of size k That is, for all datasets D, D' such that $|D - D'| \leq k$ and all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$*

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(k\varepsilon) \Pr[\mathcal{M}(D') \in \mathcal{S}],$$

where the probability space is over the coin flips of the mechanism $\mathcal{M}$

2.9.4 Differentially Private Mechanisms

Unlike k-Anonymity [1], differential privacy [8, 44] is a property of algorithms, and not a property of data. That is, we can prove that an algorithm satisfies differential privacy ; to show that a dataset satisfies differential privacy, we must show that the algorithm which produced it satisfies the above definition 2..5.

Definition 2..6 *A randomized algorithm which satisfies differential privacy property 2..5 is called privacy mechanism.*

Consequently, a privacy mechanism is essentially a randomized which quantifies the randomness producing a noise result of the original input that one would have expected from a non-private algorithm. This general method contains a number of techniques, and we will investigate here some versions for numerical statistical methods.

2.9.4.1 Randomized Response

Differential privacy builds conceptually on a prior method known as randomized response which is a research method introduced in the middle 60's [38]. Here, the key idea is to introduce a randomization mechanism that provides plausible deniability. A simple example especially developed in the social sciences [38, 46] to collect statistical information about embarrassing or illegal behavior, captured by having a property P. Study participants are told to report whether or not they have property P as follows:

1. Flip a coin.
2. If tails, then respond truthfully.
3. f heads, then flip a second coin and respond "Yes" if heads and "No" if tails.

"Privacy" comes from the plausible deniability of any outcome; in particular, if having property P corresponds to engaging in illegal behavior, even a "Yes" answer is not incriminating, since this answer occurs with probability at least 1/4 whether or not the respondent actually has property P. Accuracy comes from an understanding of the noise generation procedure (the introduction of spurious "Yes" and "No" answers from the randomization): The expected number of "Yes" answers is 1/4 times the number of participants who do not have property P plus 3/4 the number having property P . Thus, if p is the true fraction of participants having property P , the expected number of "Yes" answers is $(1/4)(1-p)+(3/4)p = (1/4)+p/2$. Thus, we can estimate p as twice the fraction answering "Yes" minus 1/2, that is, 2((1/4) + p/2) − 1/2.

Differential privacy formalizes how we define, measure, and track the privacy protection afforded to an individual as functions of factors like randomization probabilities and the number of times surveyed. In this example, there is a parameter which is the probability that the true response is recorded. If it is very likely that the true response is recorded, there is less privacy protection. Conversely, if it is unlikely that the true response is recorded, more. It is also clear that, regardless of the probability, if an individual is surveyed multiple times, then there will be less protection, even if their answer is potentially randomized every time.

2.9.4.2 Laplace Mechanism

Before analyzing the Laplace Mechanism, we introduce the concept of l_1 sensitivity which intuitively captures the maximum effect that a individual single record may cause to the output of the function that we want to estimate over the dataset. Using the definitions 2..1 2..2 we define the l_1 sensitivity between two datasets D, D'

Definition 2..7 *(l_1 sensitivity [44, 46]) The l_1-sensitivity of a function $f : \mathbb{N}^{|\mathcal{U}|} \rightarrow \mathcal{R}^k$ is:*

$$\Delta f = \max_{D,D' \in \mathbb{N}^{|U|}} \|f(D) - f(D')\|_1$$

Now, we will examine one of the classical techniques for differential privacy. The Laplace Mechanism takes a deterministic function of a database and adds noise to the result. Much like randomizing the response to a binary question, adding noise to continuous valued functions provides "plausible deniability" of the true result and hence, privacy for any inputs into that computation. Initially, we must introduce the Laplace Distribution:

Definition 2..8 *(Laplace Distribution [46]) A random variable X is distributed as per the Laplace distribution L(μ, b) centered at θ with scale b if its probability density function is:*

$$\text{Lap}(x \mid \theta, b) = \frac{1}{2b} \exp\left(-\frac{|x-\theta|}{b}\right), x \in \mathbb{R}$$

We will now define the Laplace Mechanism

Definition 2..9 *(Laplace Mechanism [46]) Given any function* $f : \mathbb{N}^{|\mathcal{U}|} \rightarrow \mathcal{R}^k$ *the Laplace mechanism is defined as:*

$$\mathcal{M}_L(x, f(\cdot), \varepsilon) = f(x) + (Y_1, \ldots, Y_k)$$

where Y_i *are i.i.d. random variables drawn from Lap(*$\Delta f/\varepsilon$*).*

2.9.4.3 Exponential Mechanism

The exponential mechanism [46, 51] can be used to provide differentially private answers to queries whose responses aren't numeric. For instance "what colour of eyes is most common?" or "which town has the highest prevalence of cancer?". It is also useful for constructing better mechanisms for numeric computations like medians, modes, and averages.

The exponential mechanism was designed for situations in which we wish to choose the "best" response but adding noise directly to the computed quantity can destroy its value, such as setting price in an auction, where the goal is to maximize revenue, and adding a small amount of positive noise to the optimal price (in order to protect the privacy of a bid) could dramatically reduce the resulting revenue.

The exponential mechanism is the natural building block for answering queries with arbitrary utilities (and arbitrary non-numeric range), while preserving differential privacy. Given some arbitrary range $\mathcal{R}$, the exponential mechanism is defined with respect to some utility function $q : \mathbb{N}|\mathbf{U}| \times \mathcal{R} \rightarrow \mathbb{R}$ which maps database/output pairs to utility scores. Intuitively, for a fixed database x, the user prefers that the mechanism outputs some element of R with the maximum possible utility score. Note that when we talk about the sensitivity of the utility score $q : \mathbb{N}|\mathbf{U}| \times \mathcal{R} \rightarrow \mathbb{R}$ we care only about the sensitivity of q with respect to its database argument; it can be arbitrarily sensitive in its range argument [46]:

$$\Delta q \equiv \max_{r \in \mathcal{R}} \max_{D, D' : \|D - D'\|_1 \leq 1} |q(D, r) - q(D', r)|$$

The intuition behind the exponential mechanism is to output each possible $r \in R$ with probability proportional to $\exp(\varepsilon q(D, r)/\Delta q)$ and so the privacy loss is approximately [52]:

$$\ln\left(\frac{\exp(\varepsilon q(D, r)/\Delta q)}{\exp(\varepsilon q(D', r)/\Delta q)}\right) = \varepsilon[q(D, r) - q(D', r)]/\Delta q\Big) \leq \varepsilon$$

Definition 2..10 *(The Exponential Mechanism [46,52]) The exponential mechanism* $M_E(D, q, \mathcal{R})$ *selects and outputs an element* $r \in R$ *with probability proportional to* $\exp\left(\frac{\varepsilon q(D,r)}{2\Delta q}\right)$

When the range of exponential mechanism is super-polynomially large in the natural parameters of the problem, it can define a complex distribution over a large arbitrary domain and so it may not be possible to implement it efficiently.

2.9.4.4 Median Mechanism

The median mechanism is a differentially private mechanism answering predicate queries $f_1, f_2, ..., f_k$ on the fly where k could be large, even super-polynomial. Without future knowledge queries, this mechanism determines the appropriate correlations between different output perturbations. Moreover, the median mechanism is the first interactive mechanism better than the Laplace mechanism, and its performance is close to the best possible even in the non-interactive setting [53].

Classifying queries as "easy" or "hard" with low privacy cost is the main concept of mechanism, for the "hard" queries there is an upper bound to $\mathcal{O}(\log k \cdot \log|X|)$ because of VC (Vapnik-Chervonekis) argument dimension [54] and the constant factor reduction of the number of databases consistent with the mechanism's answers, every time we answer a "hard" query [53].

The procedure of median mechanism can be explained in the following algorithm as described in [53]:

Algorithm 2 Median Mechanism

procedure MM(X,$f_1, f_2, ..., f_k$)
  Initialize $C_0 = \{$ databases of size m over $X\}$.
  for all queries $f_1, f_2, ..., f_k$ **do**
    Define r_i and let $\hat{r}_i = r_i + \text{Lap}\left(\frac{2}{cn\alpha}\right)$
    Let $t_i = \frac{3}{4} + j \cdot \gamma$, where $j \in \left\{0, 1, \ldots, \frac{1}{\gamma}\frac{3}{20}\right\}$ is chosen with probability proportional to 2^{-j}
    if $\hat{r}_i \geq t_i$ **then**
      set a_i to be the median value of f_i on C_{i-1}
    if $\hat{r}_i < t_i$ **then**
      set a_i to be $f_i(D) + \text{Lap}\left(\frac{1}{n\alpha'}\right)$
    if $\hat{r}_i < t_i$ **then**
      set C_i to the databases S of C_{i-1} with $|f_i(S) - a_i| \leq c/50$
    else
      $C_i = C_{i-1}$
    if $\hat{r}_j < t_j$ for more than $20m \log|X|$ values of $j \leq i$ **then**
      Halt and report failure

The median mechanism makes use of several additional parameters which are set in [53]

$$m = \frac{160000 \ln k \ln \frac{1}{\epsilon}}{\epsilon^2}$$

$$\alpha' = \frac{\alpha}{720 m \ln |X|} = \Theta\left(\frac{\alpha\epsilon^2}{\log |X| \log k \log \frac{1}{\epsilon}}\right)$$

$$\gamma = \frac{4}{\alpha'\epsilon n} \ln \frac{2k}{\alpha} = \Theta\left(\frac{\log |X| \log^2 k \log \frac{1}{\epsilon}}{\alpha\epsilon^3 n}\right)$$

The α' can be thought of as our "privacy cost" as a function of the number of queries k.

3. Disk based Clustering for Big Data Anonymization

Big data have become a reality with the new millennium. Almost any human activity leaves a digital trace that is collected and stored by someone (sensors of the Internet of Things, social apps, machine-to-machine communication, mobile video, etc.). As a result, data from several different sources are available, and they can be merged and analyzed to generate knowledge. Therefore, in this day and age, big data is the big star and they are important for both research and business. For instance, companies use big data in their systems to improve operations, provide better customer service, create personalized marketing campaigns and take other actions that, ultimately, can increase revenue and profits.

Big data provides valuable insights into customers that companies can use to refine their marketing, advertising, and promotions to increase customer engagement and conversion rates. Businesses that use it effectively hold a potential competitive advantage over those that don't because they can make faster and more informed business decisions. Both historical and real-time data can be analyzed to assess the evolving preferences of consumers or corporate buyers, enabling businesses to become more responsive to customer wants and needs.

However, as we said above big data are used from scientists and researchers. For example, medical researchers for identifying disease signs and risk factors and doctors for helping diagnose illnesses and medical conditions in patients. In addition, a combination of data from electronic health records, social media sites, the web and other sources gives healthcare organizations and government agencies up-to-date information on infectious disease threats or outbreaks. Big data are characterised by their huge volume, their velocity and their variety.

For a long time now, organizations and companies have been investing in developing solutions for implementing systems with enough computing power to process large volumes of data. Even though big data are extremely valuable in many fields, they increasingly threaten the privacy of individuals on whom they are collected. So, privacy is significant; however,

big data are complex and contain so much volume that it is difficult to manage them all in the main memory of an operating system. Indeed, the complexity is quite high, while at the same time, the data types are many. For example, some data is stored in a traditional relational database, but data such as documents, various files, videos, and images are unstructured.

From the above paragraphs, the conclusion is born that so much information in different formats makes it impossible to manage data in traditional ways and, more specifically, with algorithms and programs that use only the main memory of a machine. Therefore, big data anonymization requires smart and optimal management of all resources. For a privacy model to be useful for big data, it must be compatible with the volume, the velocity, and the variety of this kind of data [55].

So, we were led to the design and implementation of a new algorithm for Amnesia Anonymization Tool which will be adapted to the big data and will utilize the hard drive so that the main memory is not overloaded. Using a clustering logic [56], the data are loaded in small quantities on the main memory where they can be anonymized. The privacy model that perfectly fits on clustering approach is k-anonymity [1, 2] which is used in our algorithm and essentially applying local recording [21] . So the main problem now is which clustering method we will use in order to split the data with the least computational cost using the hard drive for the main volume of the data and the main memory for the procedure. Let us explore some popular cluster approaches in anonymization field.

3.1 Related Clustering Approaches

J.-W. Byun et al. [12] proposed the greedy k-member clustering algorithm for k-anonymization using clustering. This algorithm works by first randomly choosing an initial record r to start building a cluster and consecutively picking and adding more records to the cluster such that the attached records incur the least information loss within the cluster. Once the number of records in this cluster reaches k, the algorithm chooses a new record that is the furthest from r, and repeats the same process to create the next cluster. Once fewer than k records are not yet assigned to any clusters yet, this algorithm then individually attaches these records to their respective closest clusters. However this algorithm is slow with time complexity $O(n^2)$ and sensitive to outliers.

Loukides and Shao [57] proposed another greedy algorithm for k-anonymization using the clustering concept. The algorithm forms one cluster at a time. Nevertheless this algorithm picks the initial records for each cluster randomly. Furthermore, when creating a cluster, this algorithm continues choosing and attaching records to the cluster until the information loss of the cluster exceeds a user-defined threshold. Consecutively, if the number of records in this cluster is fewer than k, the whole cluster is deleted. With the help of the user-defined threshold, this algorithm is less sensitive to outliers. The main disadvantage of this algorithm is the selection of a proper value for the user-defined threshold. Moreover the deletion of many clusters may cause a considerable information loss. The time complexity of this algorithm is $O\left(\frac{n^2 \log(n)}{c}\right)$ where c is the average number of records in each cluster.

Chiu and Tsai [58] proposed a weighted feature C-means clustering algorithm for k-anonymization. This algorithm attempts to form all clusters simultaneously by first randomly picking $\lfloor \frac{n}{k} \rfloor$ initial records for the clusters. This algorithm then assigns all records to their respective nearest clusters, and updates feature weights to minimize information loss. This step is iterated until the assignment of records to clusters stops changing. Since some clusters might contain fewer than k records, a final step is necessary to merge those small clusters with large clusters to meet the constraint of k-anonymity. The time complexity of this algorithm is $O\left(\frac{cn^2}{k}\right)$, where c is the number of iterations needed for the assignment of records to clusters to converge.

Zhu and Ye [59] proposed a density-based clustering method for k-anonymization. The density of a record is defined as the inverse of the sum of distance from this record to each of its $k-1$ nearest neighbors. A cluster is created by the record with the lowest density and its $k-1$ nearest neighbors. This step is repeated until there are fewer than k records to be assigned to any clusters. Eventually, this algorithm individually assigns these remaining records to their respective nearest clusters.

Lin and Wei [60] proposed a two-stage algorithm, called OKA. During the first stage, the algorithm clusters data using the K-means algorithm, but it only runs one iteration of the K-means algorithm. During the second stage, it removes records from those clusters with more than k records (called the shrinking clusters) and attach them to those clusters with fewer than k records (called the growing clusters) such that each cluster ultimately contains no fewer than k records. The time complexity of this algorithm is $O\left(\frac{n^2}{k}\right)$.

Zheng et al. [61] proposed a clustering-based k-anonymity algorithm that considers the overall distribution of quasi-identifier groups in a multidimensional space. The algorithm first selects randomly an initial record r as a centroid of the first cluster and adds the $k-1$ closest records to it, in order to form the first cluster. Then the algorithm selects the record which has the biggest distance between itself and the initial centroid and set it to the second centroid. The i^{th} centroid is created by in the same way, based on the distance between the i^{th} record and all the existed centroids. After each centroid creation step, the algorithm attaches the k-1 closest records to the centroid to form the clusters. The algorithm iterates the remaining records and adds each record to the closest cluster, i.e., having the smallest distance with its centroid. All the clusters created contain k records at the end of this process if there are ungrouped records.

3.2 Clustering algorithm description

Our anonymity clustering algorithm is a combination of the previous clustering approaches and mainly uses the hybrid clustering method proposed by Lin et al. [62]. This method essentially combine the One-pass K-means (OKA) [61] and k-member algorithm for anonymization [12]. The basic ideas of this hybrid method are using OKA to reduce the total information loss, and using the k-member algorithm to reduce the variance of information loss among clusters. Before analyzing the disk-based clustering algorithm in detail, let us take a

Direct-Identifiers		Quasi-Identifiers				
Name	Family Name	Age	Gender	Salary	Date of Death	Disease ICD Code
Conan	Curbeam	89	male	5422	11/05/1957	ICD9:7048
Yovonnda	Leos	43	female	1280	04/05/1980	ICD10:O91213
Keith	Lowder	28	male	7785	18/01/1980	ICD9:94158
West	Entel	76	male	2374	30/11/1964	ICD10:V983
Humfrey	Venner	30	male	4214	12/11/1979	ICD9:3530
Zulema	Aliberti	4	female	3012	10/08/1988	ICD10:S32313K
Oswell	Hietala	59	male	895	09/06/2004	ICD9:59970
Shizue	Ornellas	22	female	1624	14/08/1983	ICD9:0579

Table 21: Big Data model

look at the structure of the big data that the algorithm will try to anonymize.

3.2.1 Data Model

This thesis focuses on datasets or databases with both numerical and categorical attributes. The examined data model relates to a database D with total records $|D|$. Every record $r \in D$ contains a set of columns $A = (a_1, a_2, ..., a_d)$ the quasi-identifiers are selected by the user and described by the set $QI \subseteq A$. The attacker has a partially knowledge of QI, so the main purpose is to anonymize each quasi-identifier column. This type of data model may contains millions of records and substantial amount of columns so its management it is difficult in the main memory, which very limited in relation to the volume of the data.

The use of databases representing the above structure model is often encountered in daily life. For instance, the data model is capable to describe banking transactions, surveys, medical datasets, demographics etc. The table 21 illustrates the examined data model. The attribute which contains the ICD code of a disease is normally sensitive. However, the user can set it as a quasi-identifier for simplicity and more privacy protection without to perform more complicated additional procedures such as l-diversity [27].

3.2.2 Hybrid Method

Now let us move on to a detailed description of the Hybrid method [62] in which we have made some modifications to the whole procedure especially in k-member step. Moreover, we have customized the approach using the hard disk and parallel programming to be flexible enough on the big data. Therefore, the main concept is that the dataset will be loaded by the user and stored in a built-in SQLite database. After that, the data will be split into the clusters (OKA step) and will be anonymized based on k-anonymity privacy model (k-member step).

So let us delve into the details, more specifically how the data will be grouped based on the description of Byun et al [12] and Jun-Lin Lin et al [62]. Let $\mathcal{T}$ denote a set of records, which is described by m numeric quasi-identifiers $N_1, ..., N_m$ and q categorical quasi-identifiers $C_1, ..., C_q$. Let $\mathcal{P} = \{P_1, ..., P_p\}$ be a partitioning of $\mathcal{T}$, namely, $\bigcup_{i\in[1,p]} P_i = \mathcal{T}$, and $P_{\hat{i}} \cap P_{\breve{i}} = \varnothing$ for any $\hat{i} \neq \breve{i}$. Each categorical attribute $C_{i\in[1,q]}$ is associated with a distinct hierarchy tree H_{C_i} and each numeric attribute $N_{i\in[1,m]}$ is associated with a range hierarchy tree H_{Ni}. Both distinct and range hierarchy trees are used to generalize the values of a attribute as depicted in figures 25 26.

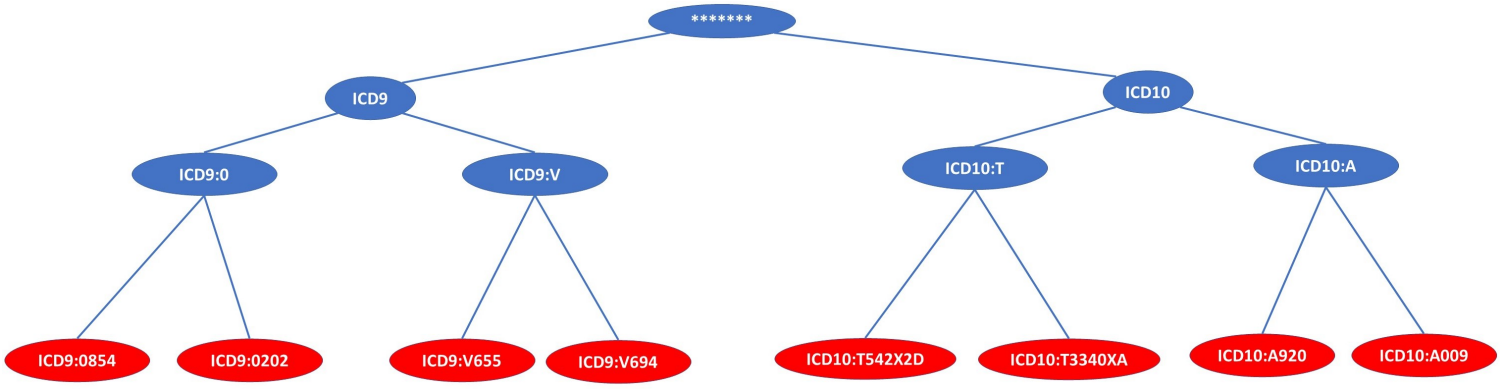

Figure 25: Distinct hierarchy

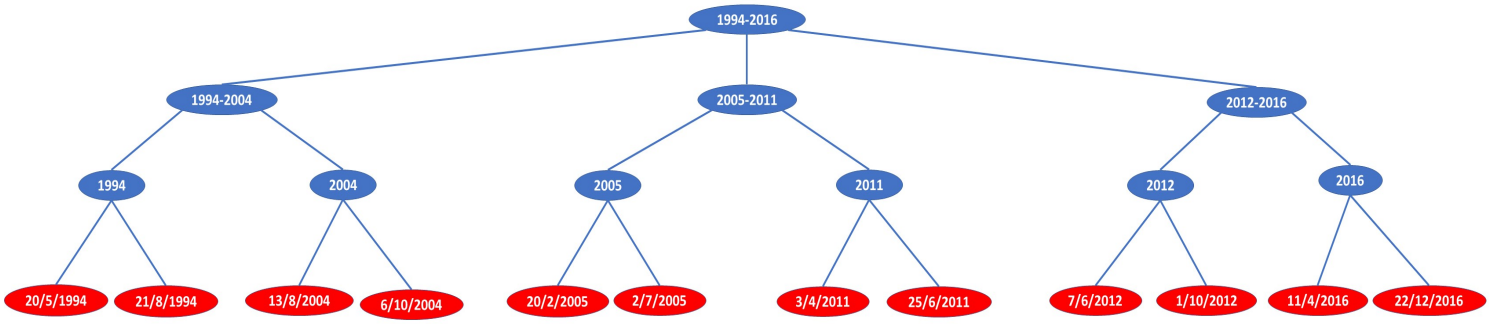

Figure 26: Range hierarchy

Each cluster-partition is represented by a centroid structure $\bar{P}$ which is basically a record whose value of attribute $N_{i\in[1,m]}$ equals $\bar{P}[N_i]$ which is the average values of the records in P, and the value of attribute $C_{i\in[1,q]}$ equals $\bar{P}[C_i]$ which is the lowest common ancestor in distinct hierarchy tree H_{Ci} of the values C_i in P's records. Thus, in this way we can implement processes for calculating the distance of a record from the cluster and updating these values when a new record is inserted into the cluster. The procedure of the distance calculation will be explained in the below section.

Regarding the cluster initialization, the algorithm randomly selects $\lfloor \frac{n}{k} \rfloor$ records with their identical in order to seed the clusters as described in [58]. Then the following questions arise, how will the clusters be initialized? What will be their size? Moreover, what will be the total number of these clusters? Knowing the number of records that exist in the database and the user-defined k we can easily find the number of blocks with size k; however, their

maximum size will be $2k-1$ when it is exceeded, they must be split down into two new partitions illustrated by 27. Splitting helps to reduce information loss during the anonymization procedure, e.g., if a cluster contains a substantial number of records, then the lowest common ancestor of categorical attributes will be near the root of the hierarchy tree producing more general values in the anonymization dataset.

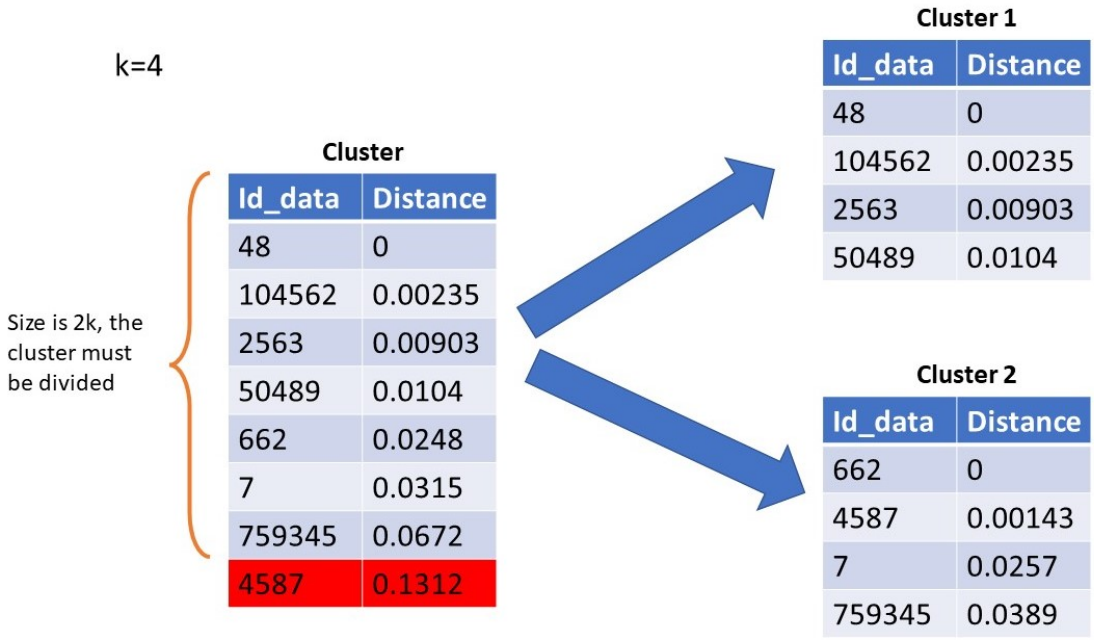

Figure 27: Splitting procedure

The clusters are essentially stored in the database, in the main memory there are only the centroids structures that represent the clusters. So basically with centroids it is not necessary to have the data records in the main memory; all the essence and functionality of the algorithm is based on centroid structure. More specifically, the clusters are stored in the hard disk in a relational table in which each record consists of the cluster-ID which is a simple number, the distance of the data record from the centroid and the data record-ID from the original data table, as a foreign key. So there is no need for a separate table for each cluster, as demonstrated in figure 28, saving disk space as well as speed because the process of creating a database table is expensive.

Additionally, we must take into account the clusters which have small size (fewer than k) when the filling process of the clusters with the original data is completed. It is almost certain that some clusters, when the access to the dataset is terminated in order to assign the original records to a partition (OKA step), some will remain fewer than k records, and therefore it will not be possible to anonymize them. So, a final step is needed to merge those small clusters with large clusters to meet the constraint of k-anonymity (k-member step) such as in [58]. In our approach the records of the problematic clusters are redistributed to the bigger clusters in order to avoid clusters with unsatisfactory size.

Finally another problem that arises is how clusters records should be stored in the main memory in order to anonymize them. Therefore, our main contribution in the algorithm is the implementation of a tree structure similar to B-tree [63] which groups the identical

Id_data	Cluster_id	Distance
592349	20264	0.006289
62865	16734	0.0000013
22	2172	0
3045	1378	0.01103
616	21	0.0184
8	15579	0.000025
285624	2141	0.014
83889	142	0.002875
⋮	⋮	⋮
57442	19200	0.0093

Figure 28: Relational database table containing all the clusters

centroids based on the total number number of clusters and the available main memory i.e. centroids that have short distances (Reminder: cetroids are also records, so it is possible the calculation of distance metrics) are grouped together so that identical or similar cluster records can be loaded into main memory without being overloaded and stored at close locations for quick access.

Below we present the algorithm's pseudocode as well as its visualization in the figure 29.

Algorithm 3 Disk-based Clustering anonymization algorithm

Input
Dataset T (set of n records)
H ($m + q$ hierarchies for each quasi-identifier)
k value for k-anonymity
Output
T' anonymized table with k

1: Let D = save T into the disk
2: Let $p = \lfloor \frac{n}{k} \rfloor$ — p number of clusters
3: Let P = initialiseClusters(D, p, k) — P centroids of clusters
4: fillClusters(P, D, k, H) — OKA step
5: removeSmallClusters(P, H, k) — k-member step
6: T' = anonymization(P, p, H, k)
7: **return** T'

3.2.3 Distance - Cost Metric

In the clustering process, the definition of distance is very critical for both the duration of the algorithm and the quality of the results. A good and intelligent choice of distance metric

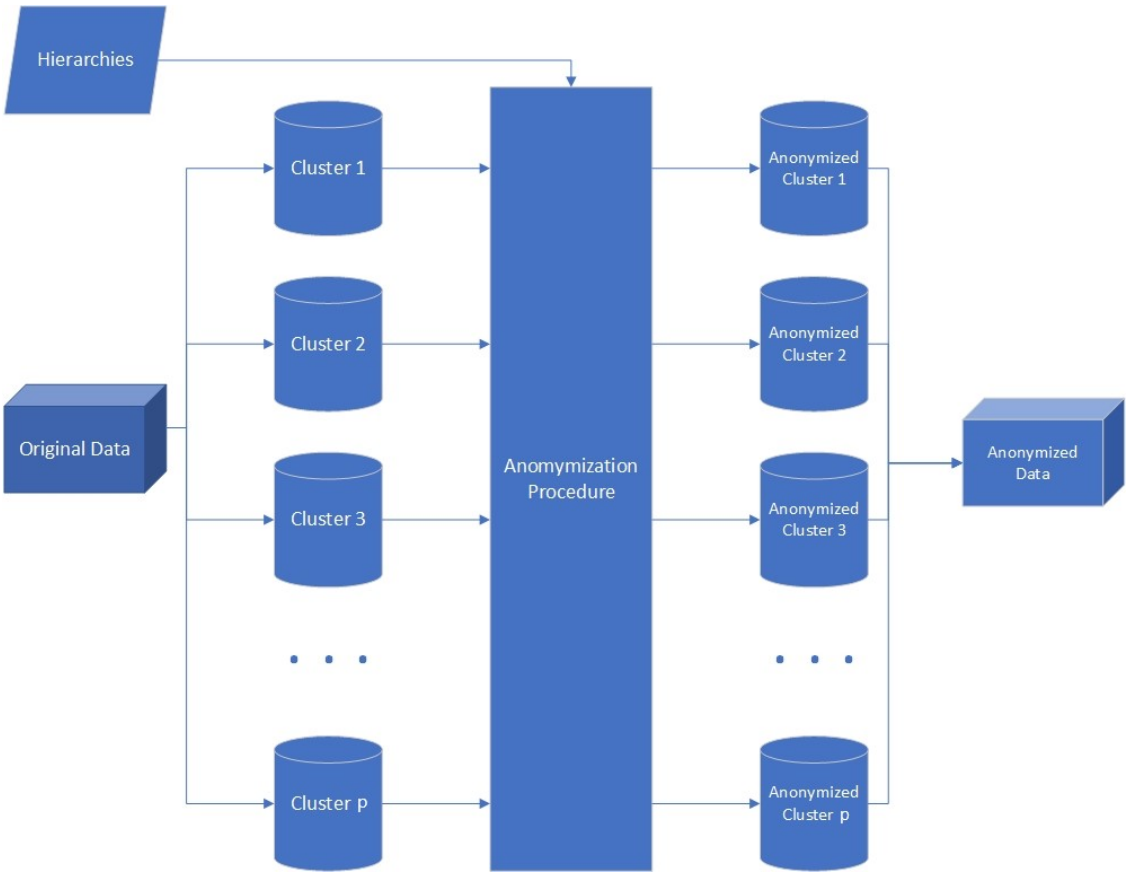

Figure 29: Disk-based clustering procedure

can lead to optimal results concerning the information loss but also in avoiding partition splitting making the process of clustering slower. In this algorithm the distance metric and the information loss are inextricably linked which makes sense since, for example, an insufficient metric can lead to the same partition two or more records which are not similar. Therefore, for the algorithm to find the equivalence class, it will need to reach high enough in the hierarchy tree, resulting in a significant information loss to satisfy the k-anonymity in this cluster.

There are several classic methods for calculating distance such as Euclidean and Manhattan. However, these are restricted to numerical quasi-identifiers only and can not handle categorical attributes e.g. Gender, Country of origin, etc. Consequently, they are not adopted in this current thesis. Essentially, we need a metric that does not just calculate a distance but a similarity between records and clusters in order to reduce the information as explained above.

- **Distance for numerical attributes:** For any numerical attribute N_i where $i \in [1, m]$ of the table T which associated with range hierarchies H_{N_i} and $\hat{N}_i, \check{N}_i$ denote the maximum and minimum values of N_i respectively, the distance for any two values $v_l, v_j \in N_i$ is defined as

$$\delta_N(v_l, v_j) = \frac{|v_l - v_j|}{\hat{N}_i - \check{N}_i}$$

- **Distance for categorical attributes:** The categorical attributes contain many discrete forms without any ordering relation; therefore the above distance metric for the numerical data cannot be applied. The only information about categorical quasi-identifiers is the generalization hierarchy which is essentially a semantic correlation of the discrete values.

  For any categorical feature C_i where $i \in [1, q]$ of the table T which with distinct hierarchies H_{C_i}, the distance for any two values $v_l, v_j \in C_i$ is defined as

$$\delta_C(v_i, v_j) = \begin{cases} 0, & v_l = v_j \\ \frac{h(H_{C_i}(v_l, v_j))}{h(H_{C_i})}, & v_l \neq v_j \end{cases}$$

  where $h(H_{C_i}(v_l, v_j))$ represents the height of the lowest common tree of v_l, v_j and $h(H_{C_i})$ denotes the total height of the distinct hierarchy tree H_{C_i}.

- **Distance between record and cluster:** To define the distance-similarity between a record and a cluster it is enough to find the distance from the corresponding centroid $\bar{P}$. However, since centroid is also a record, which contains the averages for each numeric attribute and the lowest common ancestors for each categorical attribute, it is enough to define the distance between two records.

  Let $r \in \mathcal{T}$ a record where $r = (N_1, ..., N_m, C_1, ..., C_q)$ so the distance for two records r_l, r_jT is defined as

$$\Delta(r_l, r_j) = \sum_{i=1}^{m} \delta_N(r_l[N_i], r_j[N_i]) + \sum_{i=1}^{q} \delta_C(r_l[C_i], r_j[C_i])$$

  where $r[N_i]$ represents the value of numerical attribute N_i in record r and $r[C_i]$ denotes the value of the categorical attribute C_i in record r.

  Accordingly, the distance between a centroid $\bar{P}$ of cluster $P \in \mathcal{P}$ where $\mathcal{P} = \{P_1, ..., P_p\}$, $\bigcup_{i \in [1,p]} P_i = \mathcal{T}$ and a record $r \in \mathcal{T}$ is defined as

$$\Delta(r, \bar{P}) = \sum_{i=1}^{m} \delta_N(r[N_i], \bar{P}[N_i]) + \sum_{i=1}^{q} \delta_C(r[C_i], \bar{P}[C_i])$$

  where $\bar{P}[N_i]$ is the average values of of $N_{i \in [1,m]}$ attributes of the records in P and $\bar{P}[C_i]$ equals the lowest common ancestor of the values of $C_{i \in [1,q]}$ in distinct hierarchy tree H_{Ci}.

3.3 Technical Details

Let us analyze each of the individual functions of the pseudocode 3 in more detail, bearing in mind that in each of them, there will be parallelism in order to increase the speed of the algorithm. Since, the read and write processes in the database cause a considerable delay

in the algorithm's performance and the fact that the volume of data is very large, therefore parallelism is a one-way option. Three intermediate storage areas (buffers) are also used for imports, deletions, and updates to tables in the database where a certain limit is exceeded, which is determined by the available main memory and the size of the dataset. The above database's tasks are performed in bulk where needed, significantly reducing the delay of processes when they are performed in units for each record.

- ***initialiseClusters(***$\mathrm{D}, \mathrm{p}, \mathrm{k}$***)***
  D: Original dataset stored in SQLite database
  p: The number of clusters to be initialised
  k: The value for k-anonymity

  In this procedure the clusters are initialized. Let us briefly analyze what it does. Firstly p random records are selected from the dataset. Then, the records, which have the same values in quasi-identifiers as the random ones, are loaded in the main memory; thus, some clusters may initially contain more than one record. If the set of initial records to be assigned to a cluster is greater than k, this means that this cluster is already k-anonymous. So, these records are assigned to a database table in which the whole anonymized dataset will be stored in the end. Now, taking into account the available threads of the processor, the remaining clusters that need to be initialized are equally divided to the threads that undertake their initialization. Consequently, the clusters are initialized with the help of parallelism.

- ***fillClusters(***$\mathrm{P}, \mathrm{D}, \mathrm{k}, \mathrm{H}$***)***
  P: List of centroids
  D: Original dataset stored in SQLite database
  k: The value for k-anonymity
  H: The hierarchies to be used for computing the distance and for the generalization of the original values

  In the above method, the records are assigned in clusters. The current and the previous method reflect the OKA step. Given the available memory and processor threads, a certain range of records is loaded into the main memory that is neither anonymous nor assigned to a cluster. This range of records is evenly distributed among the threads, which take on the insertion procedure into a cluster for each record. So, each thread computes the distance between an original record and every centroid in the list P in order to find the minimum distance and assign the record to the "closest" cluster. Once that happens, the corresponding centroid updates its values to be consistent with the new record described above. However, if the cluster size exceeds $2\mathrm{k} - 1$, the cluster must be split into two new ones.

  Let us analyze the split procedure, essentially half of the records that have the shortest distance remain into the cluster and its centroid is redefined according to these records. The remaining records are assigned to a new cluster and a new corresponding centroid is created based on its records and it is added in the centroid list P.

- ***removeSmallClusters(***P, k, H***)***
  P: List of centroids
  k: The value for k-anonymity
  H: The hierarchies to be used for computing the distance and for the generalization of the original values

  Since the insertion of all records into the clusters has been completed, it must be checked whether all the clusters have the appropriate size to anonymize, i.e., whether they contain at least k records as mentioned above. Using the processor's threads again for parallelism, the database ids of the problematic clusters are loaded into a list accessible to all threads. Thus, the threads take over from one cluster at a time until the list is empty, assigning its records to the nearest clusters calculating the distance with the help of the centroids and the hierarchies and deleting the cluster with a small size. The redeployment of records can be done even to another small cluster. Finally, if at least one record has been assigned into a "small" cluster then the latter is discarded by the deletion process. The current procedure is performed until the elimination of problematic partitions.

- ***anonymization(***P, k, H***)***
  P: List of centroids
  k: The value for k-anonymity
  H: The hierarchies to be used for computing the distance and for the generalization of the original values
  Output: T′, the anonymized dataset

  Let us move on to anonymize the clusters. Nevertheless, first, the tree structure must be implemented based on the set of clusters and the available main memory groups similar to centroids in a specific range in order to load the records of clusters in the main memory in nearby memory pages so that they can be accessed quickly as described above. The tree structure is depicted in the figure 30.

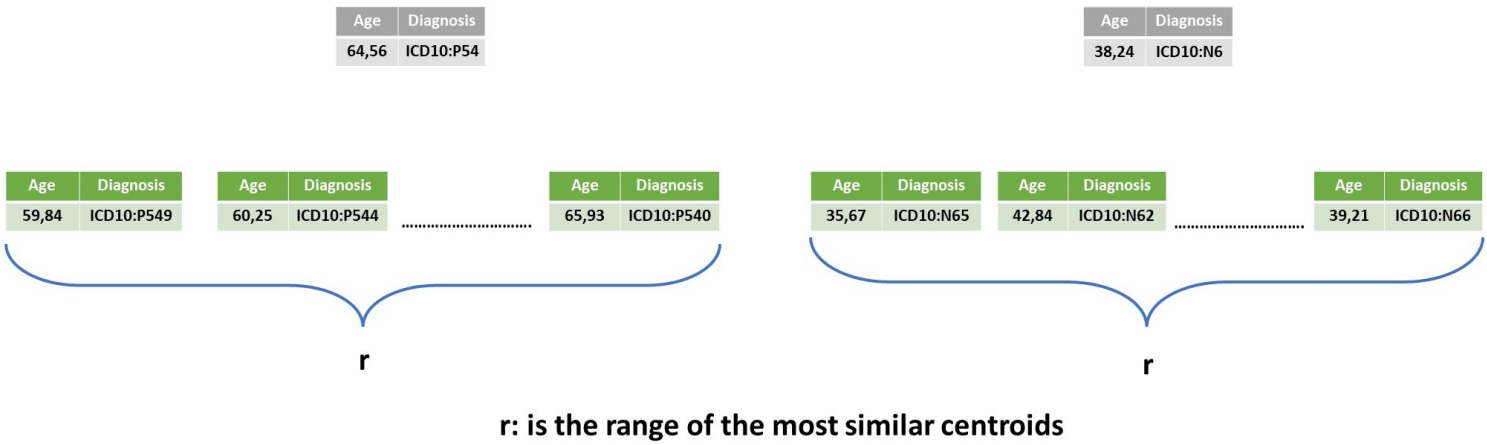

Figure 30: Tree structure which groups similar centroids

After that, the available threads of the processor will be utilized so that there is parallelism again. Therefore, each range of centroids is divided equally among the threads which then anonymize the clusters' records depending on the information present in the corresponding centroids for each quasi-identifier as illusteated in figure 31.

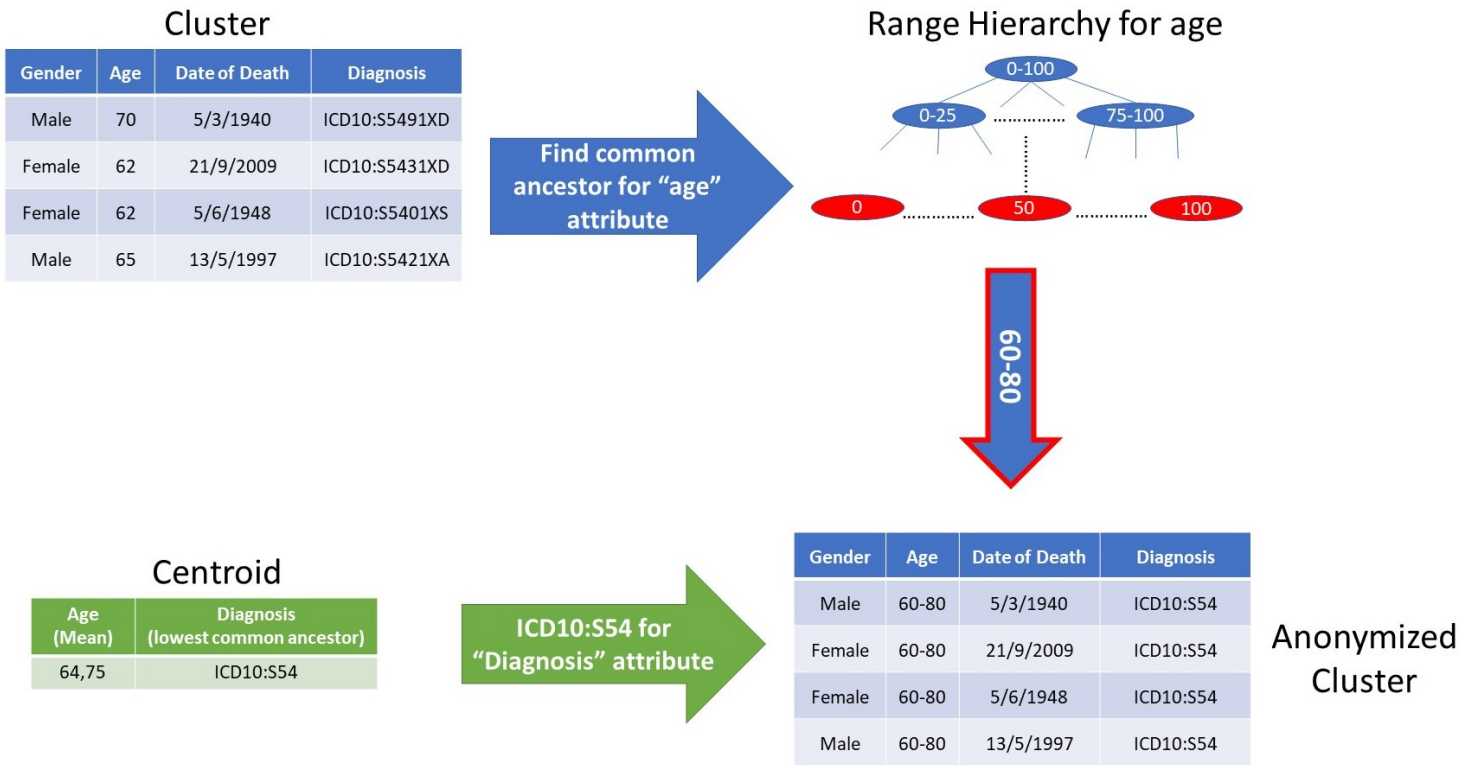

Figure 31: Cluster anonymization for 'Age' and 'Diagnosis ICD Code' quasi-identifiers

Finally, the anonymized records are inserted into the database table, where the whole anonymized dataset is stored as mentioned above. Thus, the entire original dataset was anonymized with local recording using a clustering method.

3.4 Problems and Possible Solutions

The algorithm is capable of several optimizations in order to be able to manage cases in which the total of the clusters is a few hundred thousand. This occurs when the data volume is too large and the user-defined k is too small. The main optimizations that need to be done are filling clusters with records, managing cluster splitting, and managing "small" (fewer than k records) clusters.

In these cases, which are extreme but also quite frequent, the main adjustment that must be implemented is to reduce the width of the clusters when a record is assigned into a cluster. More specifically, for each record all centroids are searched computing the distance metric in order to find the most similar cluster so, the complexity is $O(n^p)$ where the p may be increased due to cluster splits.

A satisfactory solution would be that the set of centroids to be split into smaller sets and for each of these sets to be represented by a new centroid, if we repeat the same procedure for the new centroids a tree structure is implemented. For each record that needs to be assigned to a cluster, we can compare it with the levels of the tree structure until the bottom level, where the appropriate smaller set of clusters is found. Hence, the minimum distance must be found within this particular set which is much smaller than the entire set of centroids.

3.5 Optimizations

Although we use parallelism in many cases, it is necessary to extend this method to more algorithm's functionalities. Especially, whether the choice of k, to apply the k-anonymity model, is small enough in size concerning the total number of records, and if we take into account that the processes in the database via the SQLite can not be easily parallelized, this makes the optimization of the time execution even more difficult. For example, in the initial algorithm for a dataset with one million records, the definition of k had to be at least around 500 in order for the algorithm to run in a reasonable time frame.

Eventually, what we recognized after experiments is that the execution time of the algorithm is inextricably linked to the selection of k, which is proportional to the number of records in the dataset. However, in anonymization applications, the selection of k over ten does not make much sense, let alone in the above example, which reaches between one hundred and one thousand. Consequently, it is necessary to minimize this problem, and the time execution should not depend on the user-defined k in relation to the size of the data.

Meanwhile, what makes k so important in executing the algorithm? If we refer to the pseudocode of the algorithm 3 what we observe is that a simple division calculates the number of clusters of the number of records (n) by k. So, the problem is located in the number of partitions and consequently in the space occupied by a cluster. The size of each cluster, as mentioned above, includes up to 2k-1 records. As soon as this threshold is exceeded, the cluster must be split into two more. If in the example above, with the dataset of one million records, the user sets $\mathrm{k} = 4$ then the total number of the clusters is 250,000, and if we take into account that some partitions will need to be split. At the same time there will be clusters containing records less than k, then the number of clusters is rapidly increased because it is impossible to have absolute similarity between four records per million and to have precisely 250,000 partitions with exactly four records. Thereby, some clusters will work as points of convergence and will constantly be divided where this operation is very expensive in time execution, especially if the splits are thousands.

In addition, in this case, we also have an issue in the main memory because all these clusters have a corresponding centroid, so the number of these structures will be equal to the number of clusters. Hence there is a relative overload in the main memory, which, unfortunately, we can not avoid at this moment, but there is another problem arises. To compute the distance between a record and a partition, it is necessary to access all the centroids to find the most similar cluster. The similarity search consumes substantial time, but we can

reduce it.

Finally, another problem that arose based on a large number of centroids is the creation of the tree structure in the anonymization phase. Depending on the number of clusters and the available main memory, the tree structure groups identical centroids in a specific range in order to load the clusters records in the main memory and in nearby memory page frames so that their access can be done rapidly. However, to implement this structure, it essentially makes new centroids that represent a range of similar clusters of centroids. To calculates the centroids' similarity to create the representative centroid, the whole set of centroids is continuously accessed to find the most similar. The appropriate centroid is inserted in the range centroid list then the values of the representative centroid are updated to have accuracy in the similarity-distance of the centroids. Nevertheless, when the algorithm makes continuous accesses and comparisons of similarity across the multitude of centroids, the algorithm's execution slow down rapidly, especially when the clusters are enumerated in thousands (same as centroids). As an effect, this algorithm's malfunction should be addressed.

Let us now turn in more detail at which points in the algorithm we have improved the time performance, highlighting the modifications in some functionalities and the new structures we designed to have a more optimal search and processing of the data. Our issue, as mentioned above, is when the volume of data (V) is very large and the user-defined k is too small ($V \gg k$). So, we use a relatively simple metric that we will distinguish this case to recognize in terms of implementation and apply a different approach, using a new tree structure for quick access to a specific range of clusters.

The aforementioned metric checks the order of magnitude between k and the number of records (n) by computing the k/n ratio to be greater than $10^{-}3$ and the volume of clusters to be greater than 50000. The new tree structure is very similar to the tree structure we used in the anonymization phase of the clusters. The tree is built from the bottom up, where a specific range of children is initially defined for each immediate inner node of the tree (ancestor node of initial original centroids).

The tree's leaves are the centroids that represent the clusters in the main memory stored on an array. Having defined the range of leaves-children, the centroids' array is split among the defined range, and for each partition of the array, a parent-centroid is created, which contains the average values of the centroids. We repeat the same process until we make the root as depicted in 32. This structure is used in different phases of the algorithm, while the range of children of the tree is calculated based on a percentage of the total number of clusters in the system in each phase of the algorithm.

As a result, the tree structure gives us the ability to deposit a record to a cluster without calculating the distance metric for each centroid, but it is enough to find through the tree in which partition of centroids the record has more similarities and then look for the shortest distance only to the corresponding centroids. Normally, in the creation of a tree structure, each centroid should have been checked in terms of the distance metric with all the others in order to group identical centroids; however, it is a process with a long time cost $O(n!)$.

In the end, another two important improvements were made in the algorithm in ***fillClusters*** and ***removeSmallCluster***, using the metric that analyzed above and the new tree-structure for computing the distance for each record among to the available clusters. However, in the former process, the splitting procedure significantly delayed the algorithm execution, especially when the k value was very small. So, we decided the splits to take place when all the records are set to a cluster. After that, the partitions with a size bigger than $2k-1$ are split until to reach the desired size between k and $2k-1$ using parallelism with threads. In the latter process, we load in the main memory all the records to the memory of the "small" (size less than k) clusters which are totally removed from the database. Then the records are redistributed to the remained clusters. After that, the algorithm splits the "big" (size bigger than $2k-1$) clusters as mentioned above. The described optimization occurs when the described new metric is satisfied, and the algorithm significantly accelerates despite the small increase of information loss using the tree structure.

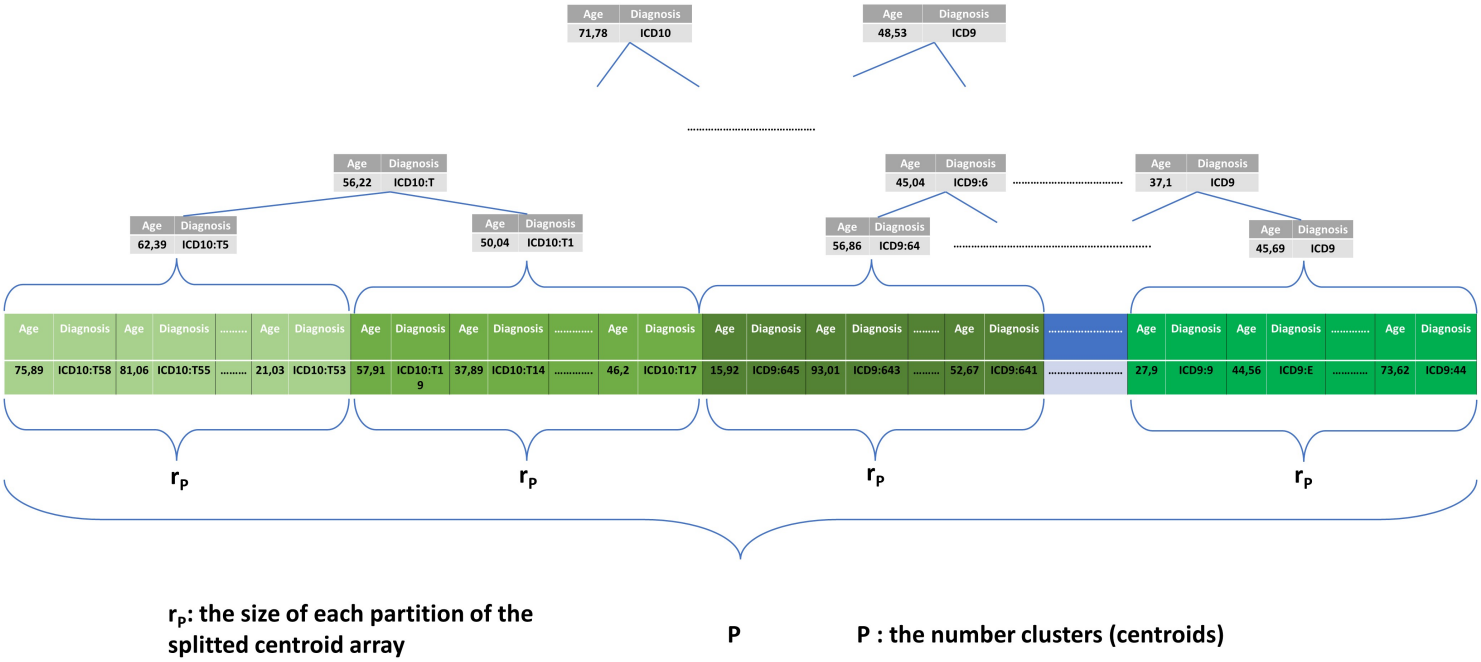

Figure 32: Tree structure which separate the centroids array in specific ranges

3.6 Evaluation

The disk-based clustering k-anonymity algorithm was evaluated in Netbeans IDE, where the Amnesia Anonymization Tool was developed. The experiments were performed using Java 18 on windows 10, 64bit Home edition, with Intel Core i7-1065G7 @ 1.30GHz 1.50GHz CPU, 16GB of RAM, and SK Hynix SSD hard disk. The dataset contains up to one million records and is depicted in figure 21 as mentioned above.

3.6.1 Execution Time

Time is a useful tool for concluding the performance of the algorithm. In particular, any difference in the execution time can be observed concerning the number of quasi-identifiers and the value of the k parameter. The following section presents the results from the different executions of the algorithm regarding its execution time.

The proposed disk-based clustering algorithm is expected to move at lower levels compared to other algorithms such as the parallel Flash, which is also present in Amnesia Anonymization Tool since our new algorithm has more complicated processes and it uses SQLite and a database stored on a hard disk to handle the data. The operations via SQLite, such as insert, delete, and the execution of queries, are really slow without the possibility of parallelism because it does not allow the database usage from two or more threads simultaneously. The execution of database operations in a second stage from one thread only and via batches, i.e., thousands of operations at the same time and not one by one, helped significantly and decisively in the algorithm's performance.

The supplied diagram 33 outlines the time execution of the algorithm using a different number of quasi-identifiers. As it is observed, the anonymization of a single quasi-identifier is much slower than the other experiments using two or more quasi-identifiers. As we mentioned above in the optimizations section, we split the algorithm into two approaches. The former is the classic approach which is slow but more efficient in terms of information loss, and the latter is more optimized regarding the time execution; however, it is weak in information loss. It is obvious that by anonymizing only the 'Date of Death' feature, the algorithm runs the classical, which achieves better performance in information loss, as we will see in the next section.

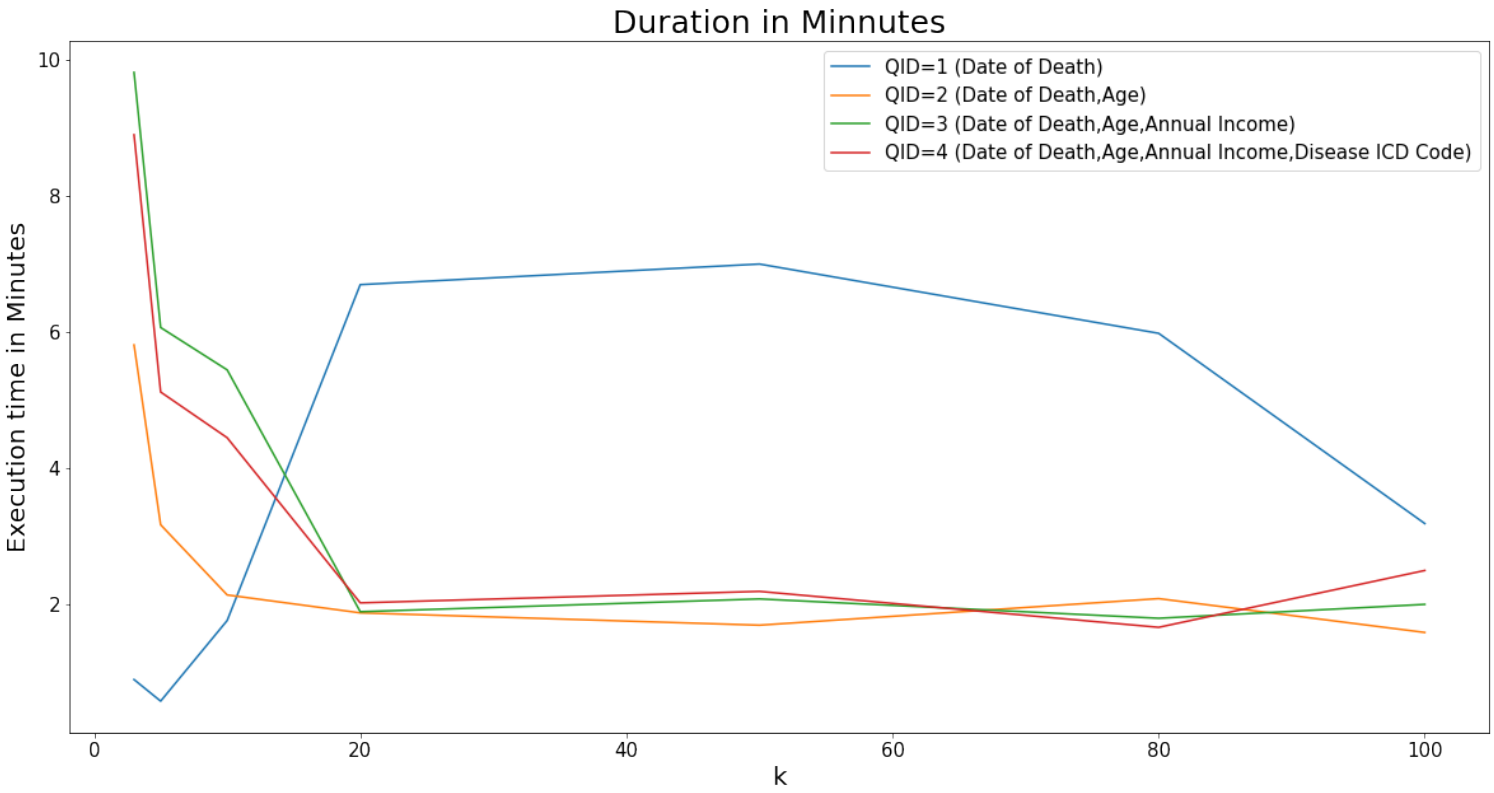

Figure 33: Execution time with varied combinations of quasi-identifiers

Another interesting note that demonstrated from the illustration 33 is the convergence of the three other cases in a given time range proves that despite the fact that the algorithm is non-deterministic the optimizations we have performed lead to the execution of the algorithm always within a specific context in order to be executed in a reasonable time frame. Of course there are other factors that affect the execution time such as e.g. the hierarchy's form.

In image 34 is depicted the time execution of the algorithm, which anonymizes only the feature "Salary" with two different hierarchies where one is a range hierarchy, and the other is a distinct hierarchy. The experiment was performed in 10 executions of the algorithm for each hierarchy and we received the average value of time. It is clear from the bar graph 34 that the execution of the algorithm with range hierarchies is more efficient in terms of time because the distance for the quasi-identifiers which have range hierarchies is more simple and fast than the others that have distinct hierarchies. This happening because of the seeking of the lowest common ancestor, which an time expensive procedure to compute the cost metric.

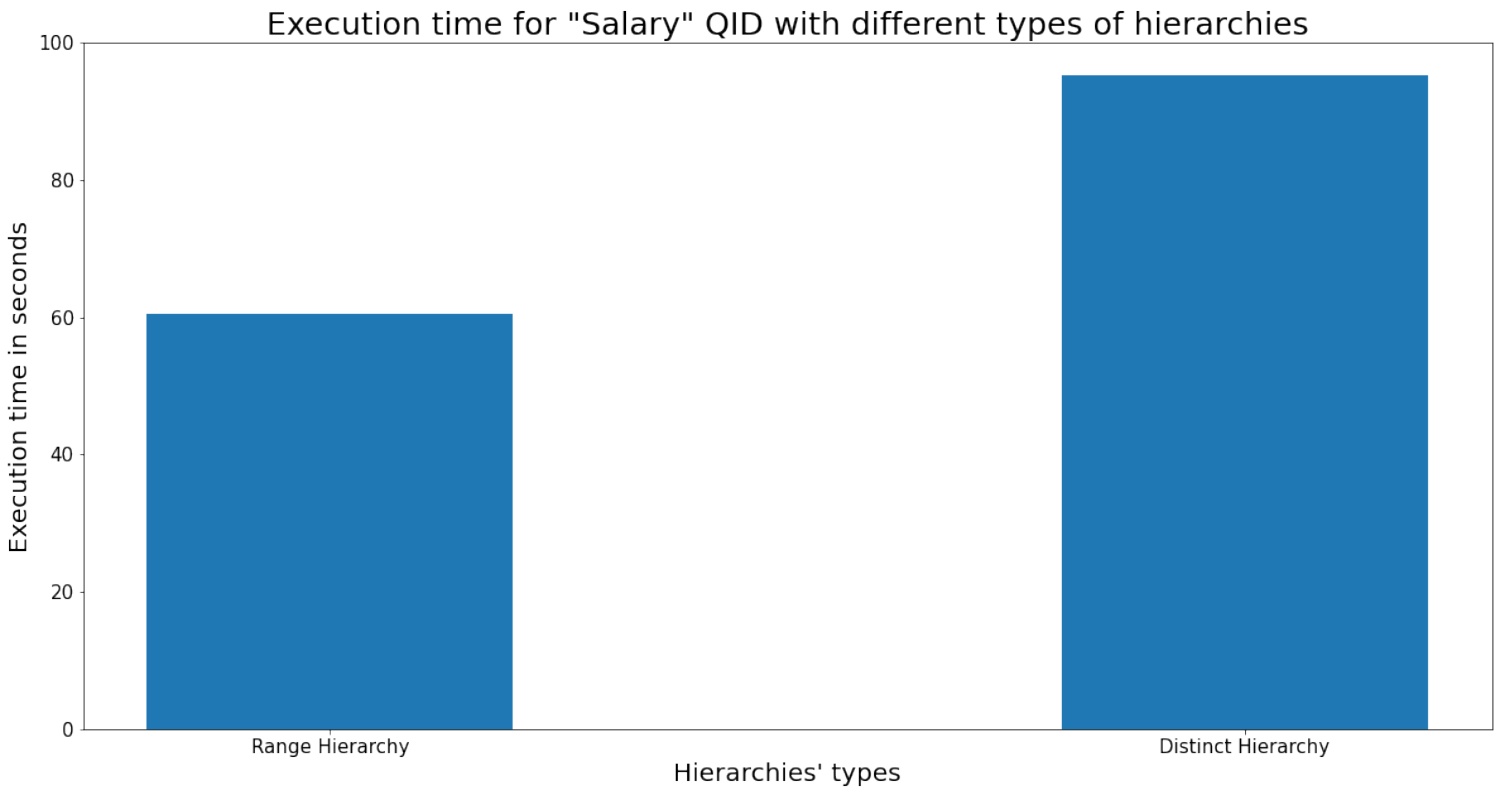

Figure 34: Mean execution time in relation to hierarchy's type

The following figure 35 presents information about the execution time of the algorithm among different data types of the quasi-identifiers. As observed, the 'Age' and 'Salary' attributes give us better results about 'Date of Death' and 'Disease ICD Code' features. This is perfectly reasonable due to the limited domain set for the first two quasi-identifiers, and in one million records, it is certain that 'Age' and 'Salary' values are repeated many times, so it is obvious that very few records are required to anonymize. More specifically, the 'Age' attribute satisfies the k-anonymity for many values of the k parameter; hence the algorithm is rapidly executed without performing the clustering approach. On the contrary, 'Date of Death' and 'Disease ICD Code' quasi-identifiers have huge domain sets, and the number of the repeated values on the dataset is minimal, especially for ICD codes.

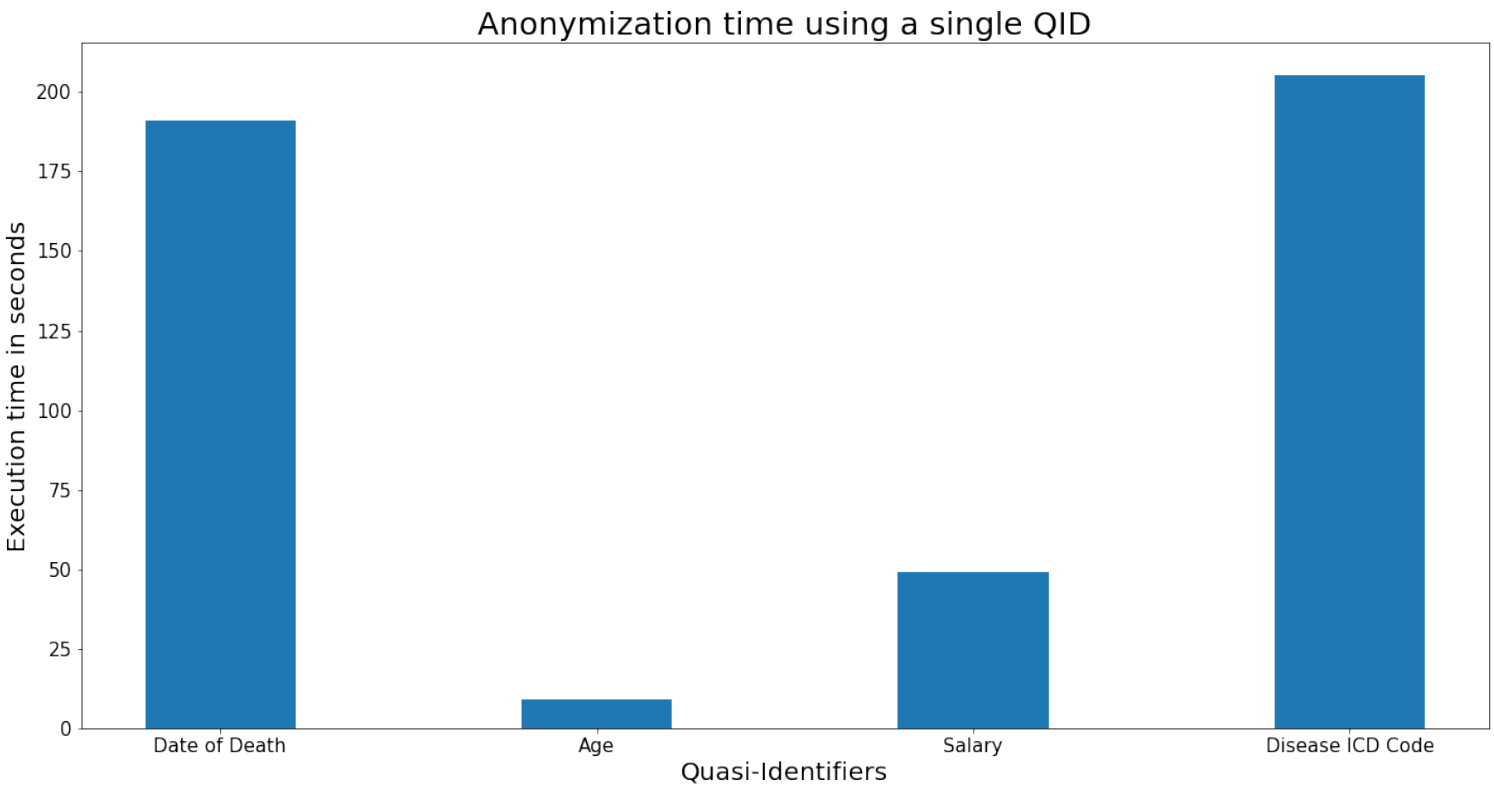

Figure 35: Execution time in relation to quasi-identifiers' data types

Finally, the last graph 36 demonstrates how the execution time of the algorithm is affected by the number of quasi-identifiers. As can be seen, the execution time is rapidly decreased between 1 and 2 quasi-identifiers due to the classic clustering approach that the algorithm follows when the QIDs=1. On the other hand, between 2 and 3 number of quasi-identifiers is observed sharply increase in the execution time, and in the end, there is a slight fall between 3 and 4 QIDs. In the current evaluation, we took the best execution times for every quasi-identifier combination from the line graph 33.

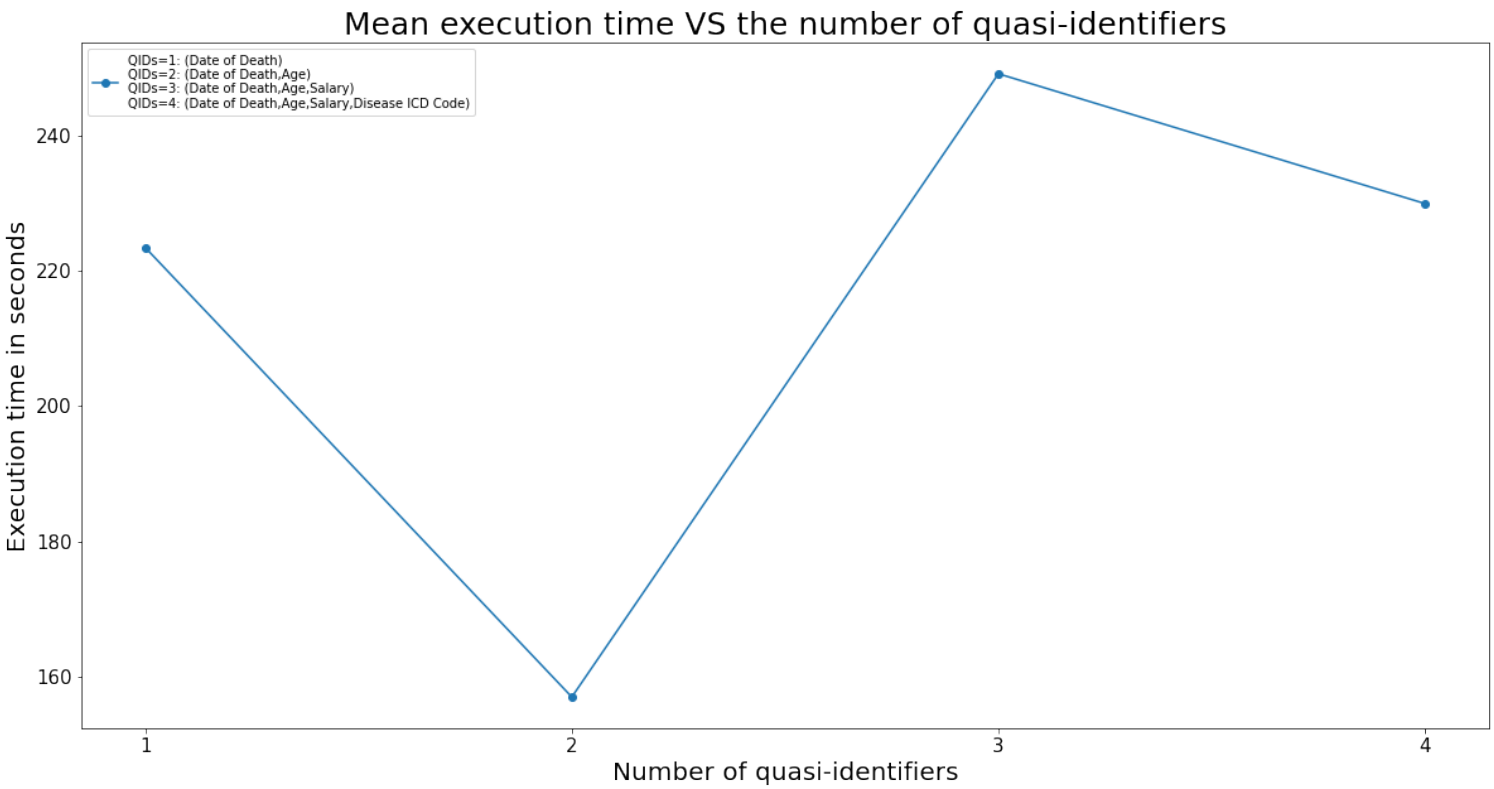

Figure 36: Mean execution time in relation to the number of quasi-identifiers

3.6.2 Information Loss

In order to perform the evaluation experiments to illustrate the information loss of the anonymized dataset we used two information loss metrcs: *Normalized Certainty Penalty (NCP)* [21] and *Total* [12].

- ***Normalized Certainty Penalty (NCP)*:** The *NCP* is one of the most popular metric that quantifies the information loss. *NCP* is a kind of algorithm to measure the degree of loss, which is efficient and easy to use. For every generalization value g, *NCP* calculates the number of leaves which are generalized by g. Then, it makes a normalization by dividing by the total number of leaves. To find the NCP of a record it is required to sum the individual *NCP* for every generalized quasi-identifier and divide by the number of quasi-identifier. Adding the *NCP* of every record, then by dividing by the total number of rows, we conclude the final *NCP* proportion for the whole dataset.
- ***Total*:** Another famous information loss metric is *Total*. *Total* takes into account the level of the generalized value in the hierarchy. More specifically, in k-anonymity, *Total* finds the level of every generalized attribute on its hierarchy and divides it by the height of the hierarchy minus 1. The ratios for every quasi-identifier are summed up and normalized by dividing by the number of quasi-identifiers.

Let us examine the experiments below and analyze the graphs of the information loss evaluation of the disk-based clustering algorithm. In the line graphs 37, 38 we can observe that proportion of information is increased when we combine 2 or more quasi-identifiers in the anonymization procedure. This is completely normal because the number of combinations for two or more columns in the dataset is usually unique or less than k. So, the algorithm has to search at a higher level in the generalization trees of quasi-identifiers to find an accepted generalized combination that satisfies the k-anonymity. Therefore, a general, logical, and unofficial rule is that the more quasi-identifiers we have in the anonymization process, the more information loss we usually obtain.

An important remark that should be noted is that in Flash algorithm we can observe much better performance in information loss metrics in many cases. However, a lot of accepted solutions in the lattice, depicted in 17, are useless due to the fact that many quasi-identifiers are generalized in a very high level and in the most cases in the root level of the generalization hierarchy and other features are not generalized because they are already k-anonymous. So, basically Flash algorithm removes the specific columns that does not satisfy k-anonymity in order to reduce information loss. For instance, in our data model 21 for quasi-identifiers 'Age' and 'Diagnosis ICD Codes' Flash produces a solution with 50% information loss in the anonymized table according *Total* metric, applying global generalization with hierarchy's root value in 'Diagnosis ICD Codes' attribute. Thus, Flash algorithm seems to have better performance than disk-based clustering algorithm in 38 in terms of information loss but, essentially, corrupts the quasi-identifiers similar to 'Diagnosis ICD Codes'.

Figure 37 the *NCP* quantifies better the information loss due to the fact that enumerates the original values that have been anonymized by a specific general value in the hierarchy tree. On the other hand the *Total* depends on the hierarchy's height. However, it is difficult to make a hierarchy with satisfactory height, for example, on the 'Age' attribute which has a limited domain mostly from 0 to 100 we can not construct a very deep and sparse hierarchy. Nevertheless, the 'Date of Death' feature has a higher and more sparse hierarchy and this is illustrated in both graphs where the information loss is substantial less than the other cases. Moreover, the figure 38 expresses that for *Total* metric 2 or more combinations of quasi-identifier make the information loss proportion to converge in specific values according to the k value. Finally, as it is described above when anonymizing the 'Date of Death' the algorithms execute the slower approach which gives fewer information loss.

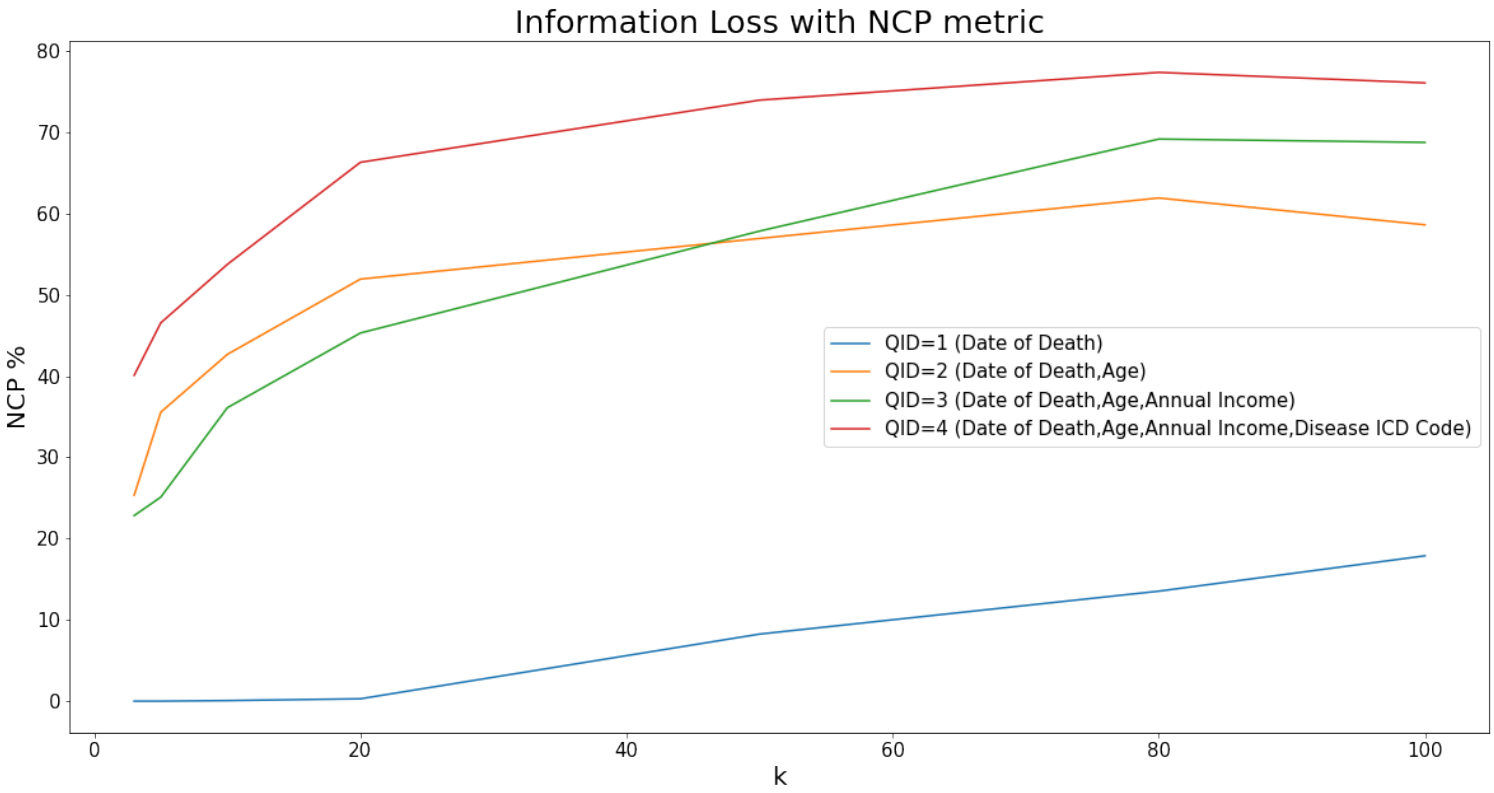

Figure 37: Information Loss using *NCP* metric with varied combination of quasi-identifiers

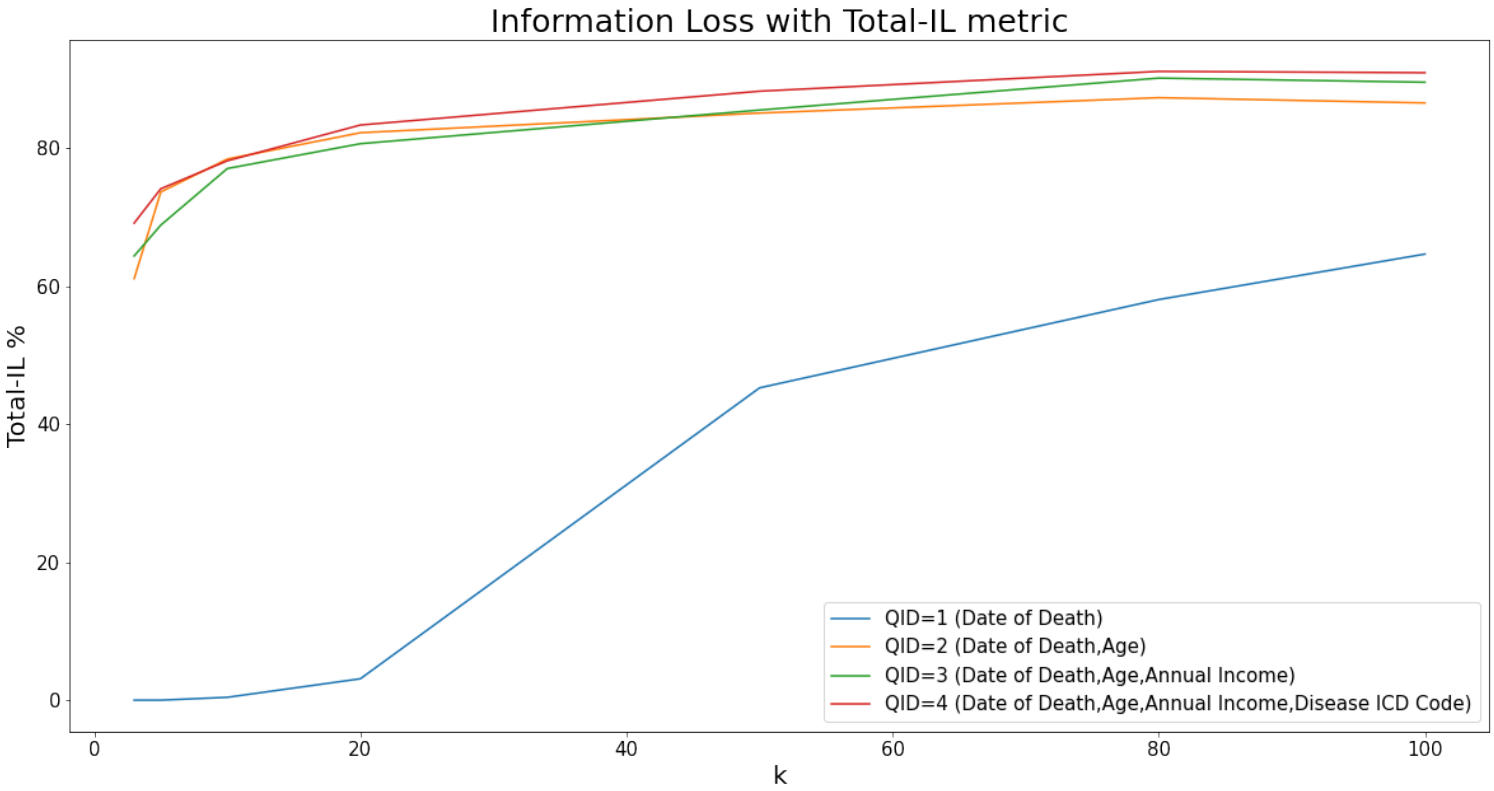

Figure 38: Information Loss using *Total* metric with varied combination of quasi-identifiers

Nonetheless, it is worth observing every quasi-identifier alone in the anonymization procedure to have a better overview and understanding of quasi-identifier hierarchies and their importance in information loss.

In figures 39 and 40 we can see the information loss for 3 from 4 quasi-identifiers for fixed $k = 100$ because for smaller values the 'Salary' quasi-identifier was not generalized at all since it satisfies the k-anonymity. Unfortunately, the 'Age' attribute has a limited domain set, and it is already anonymous for a significant number of k values without generalization. Obviously, the 'Salary' quasi-identifier has the least information loss due to the limited records that do not satisfy k-anonymity, so very few records were anonymized. Many records maintained their original salary value. On the contrary, 'Date of Death' and 'Disease ICD Codes' values' have more sparsity in the dataset, and the disk-based algorithm is obliged to generalize the values at a higher level. Moreover, the *Total* proportion is significantly larger than the *NCP* metric for the 'Date of Death' quasi-identifier. We can find this explanation in its hierarchy, which has a great height proportionally to the number of leaves in each hierarchy's generalized node.

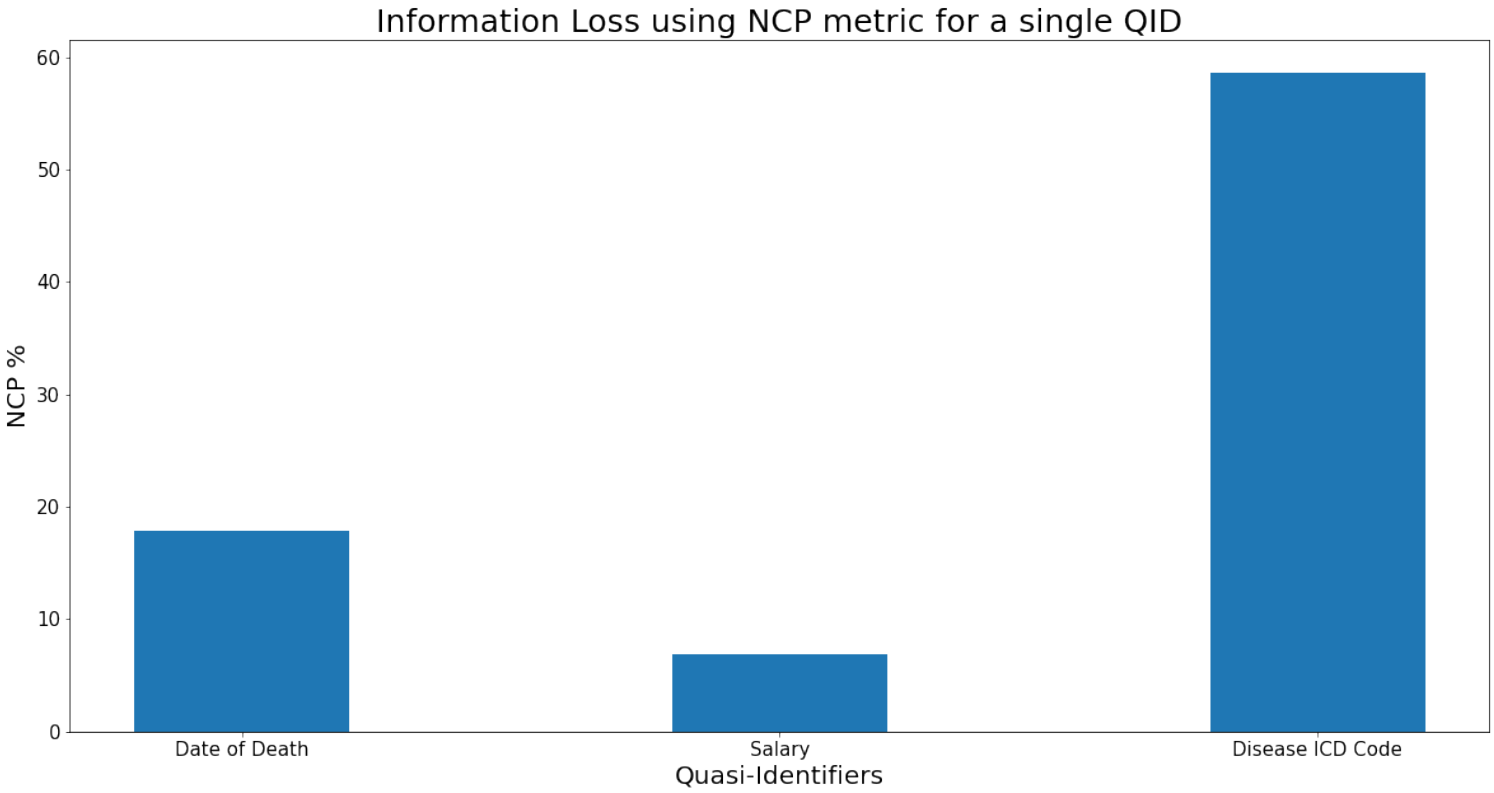

Figure 39: Information Loss using *NCP* metric for a single quasi-identifier

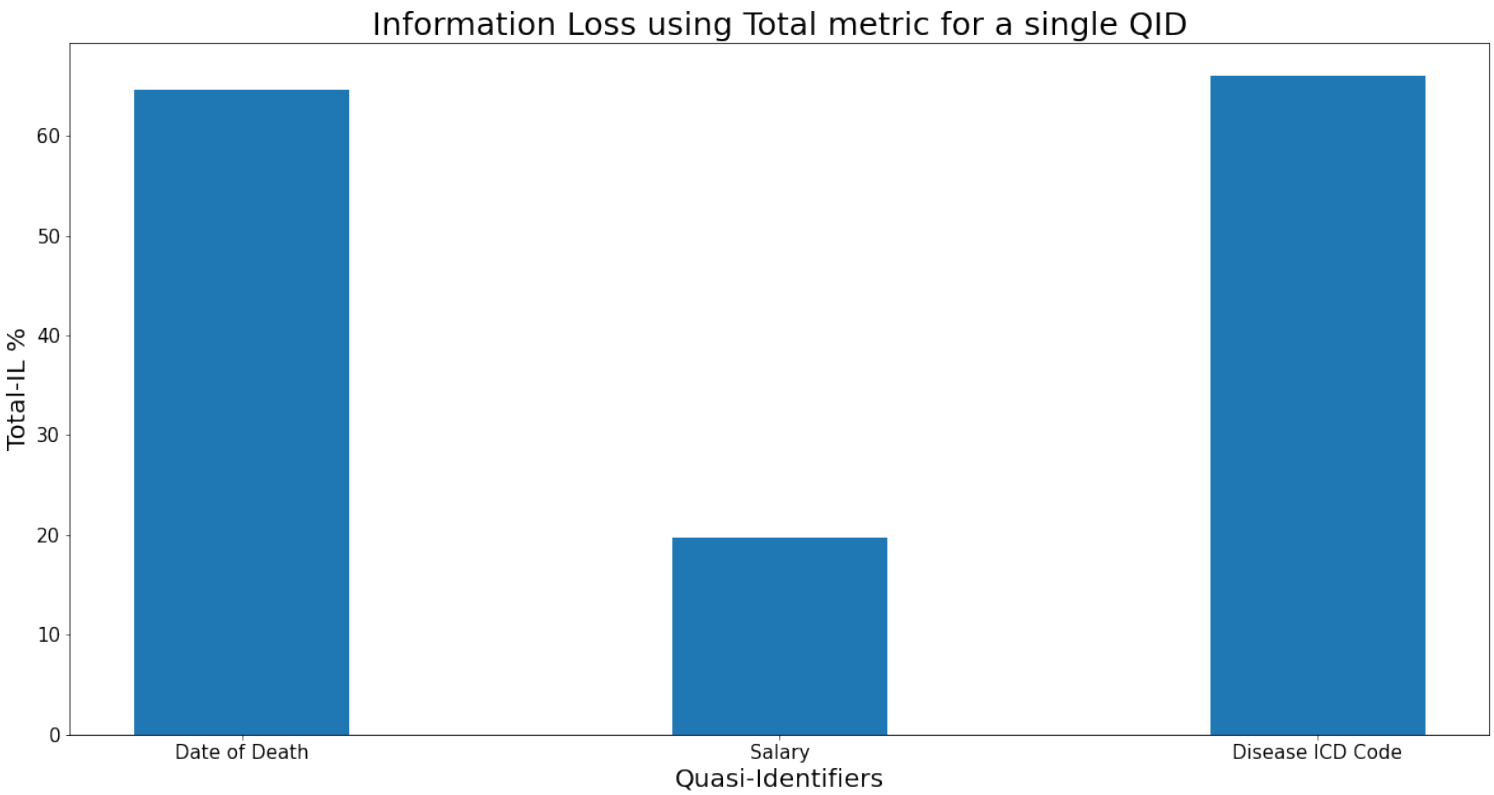

Figure 40: Information Loss using *Total* metric for a single quasi-identifier

3.7 Use case in Amnesia Anonymization Tool

Let us examine a use case scenario of our disk-based clustering anonymization algorithm in Amnesia Anonymization Tool. Summarizing the key concepts of the procedure, initially, the dataset is stored in a SQLite database because this method is focused on big data which are very difficult to handle them in the main memory. After that, the data are split in clusters having size between $[k - (2k - 1)]$. Finally, the dataset is partially anonymized with the k-anonymity privacy model and the hierarchies for the quasi-identifiers. Therefore, the solution generated by the algorithm is unique, and the main feature is that a value that exists in different records may have a different value in the anonymous dataset or even not be anonymized at all in some records (Local Recording). The following example analyzes the crucial steps that the user must follow to utilize this algorithm through Amnesia.

3.7.1 Loading Dataset

A user can upload a dataset to the Amnesia Anonymization tool with plenty of options

- **Option 1:** Clicking the orange button Amnesia shows in a modal view the available upload options as depicted in figures 41 42. The user is able to choose the source

destination of the dataset, Amnesia offers 4 loading choices:

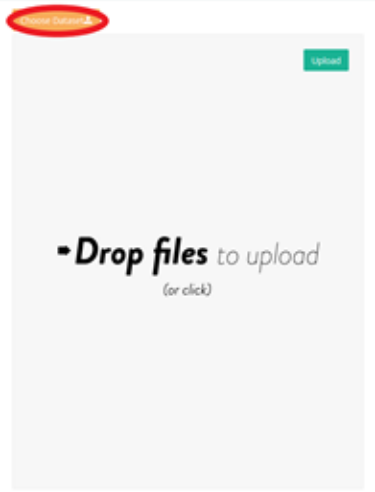

Figure 41: Amnesia Loading screen

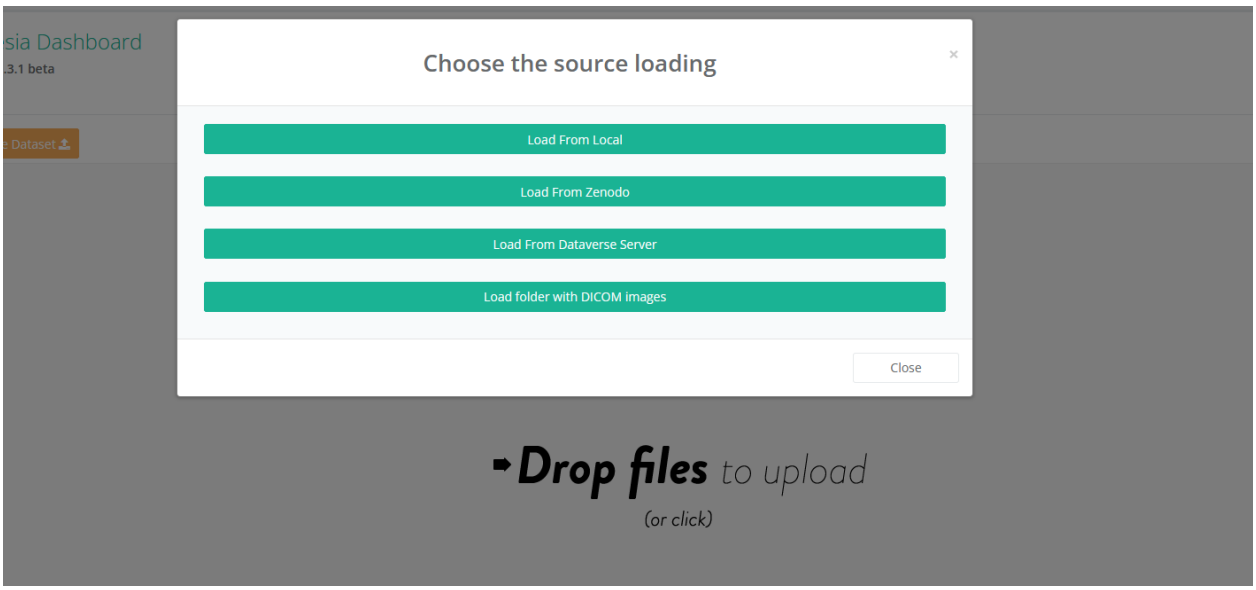

Figure 42: Amnesia sources loading

1. **Loading from Local:** The user can choose the dataset from his/her personal device.
2. **Load from Zenodo:** Loading a Dataset from Zenodo [2] repository is quite common in various software systems especially for reasearch data.
3. **Load from Dataverse Server:** A lot of institutes and research centers have installed Dataverse [64] servers in order to deposit data, papers etc. So, a scientist can easily load to Amnesia through Dataverse.

[2] https://www.eui.eu/Research/Library/ResearchGuides/Economics/Statistics/DataPortal/Zenodo

4. **Load folder with DICOM images:** A doctor or a user is able to anonymize sensitive metadata from DICOM images through Amnesia Anonymization Tool.

For simplicity in our example the dataset is loaded from a personal device. However, the user can upload the desired dataset file from many other screen positions in Amnesia application.

- **Option 2:** By simply clicking on the box or dragging it, the file that user want to upload as illustrated in figure 43.

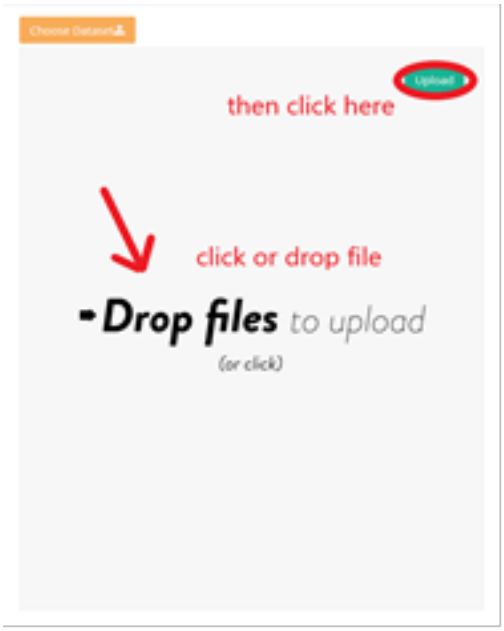

Figure 43: Amnesia drag drop to load a dataset

- **Option 3:** By clicking on the ”source” option in the left menu, then some options from the first loading case appear. However by pressing ”manage”, ignoring the warning message and then press the ”load new dataset” button and load the desired file from the local storage presented in figures 44 45.

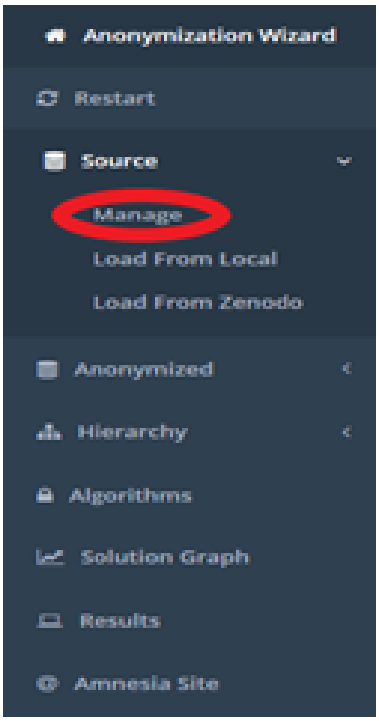

Figure 44: Amnesia loading from main menu

Figure 45: Amnesia loading dataset

After the user uploads the data file, he must specify additional information about the file. Firstly, specify the column separator as shown in the image 46. In this case, the separator is the comma. In addition, the user must specify the type of data set defined by the tool as follows:

- **Simple table**
- **Sets of values**
- **Table with a set-valued attribute**
- **Disk based simple table**

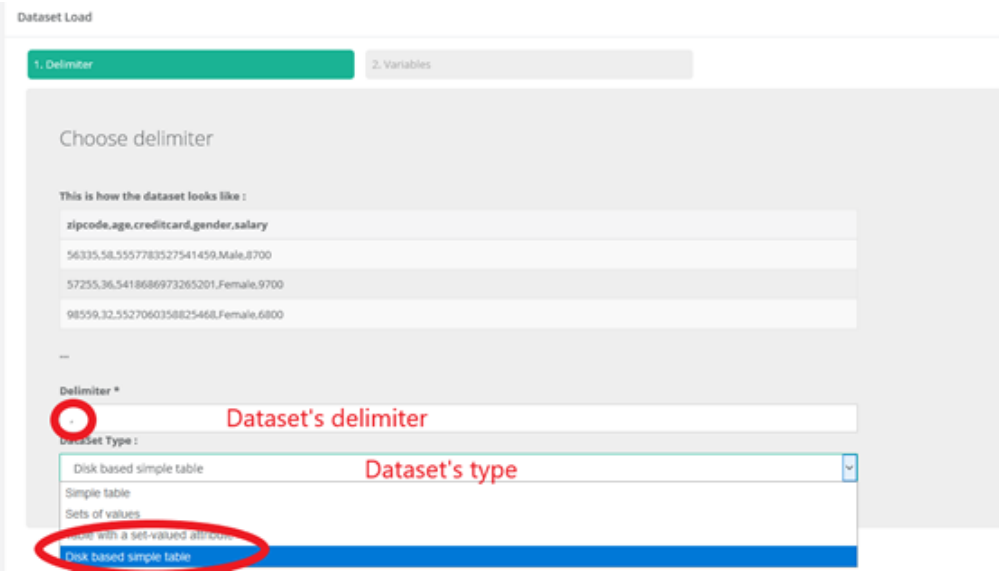

Figure 46: User-defined information about dataset

So, it is evident in our example that the dataset is loaded as a ”Disk-based simple table”. Then, Amnesia presents to the user all the columns of the data set and suggests possible data types for the attributes refers in figure 47. The user has to choose which columns he wants to load and their types. Moreover, the tool allows the user with the ”Toggle All” button to release all the columns and select each one that he/she needs in case he/she wants to abort several columns from the anonymization process.

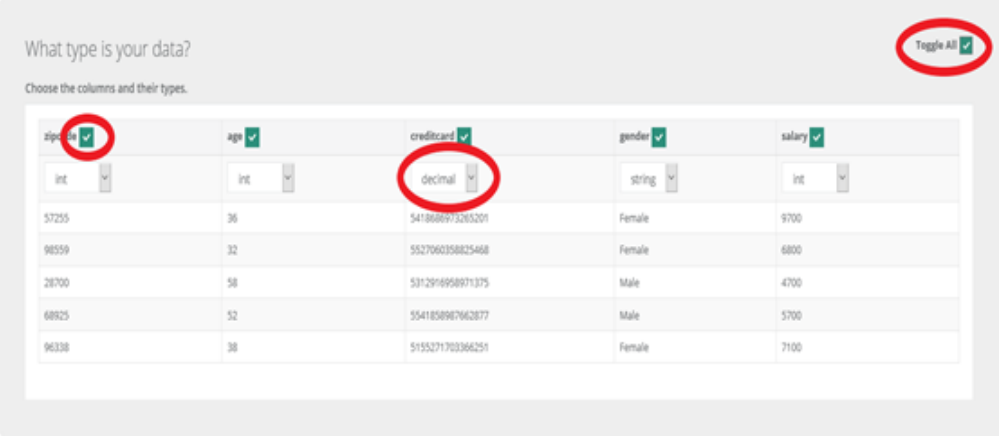

Figure 47: Choosing columns and desired data types

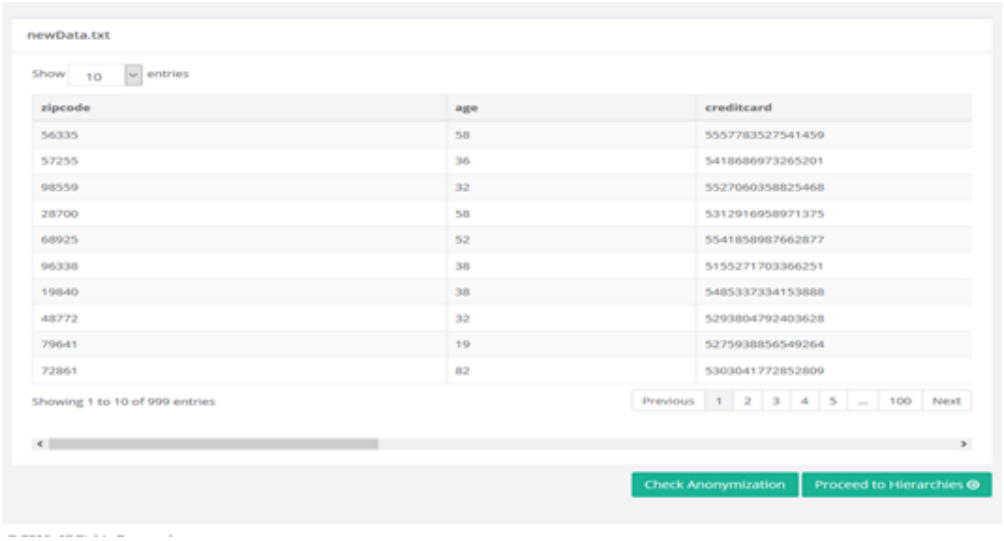

Figure 48: The loaded dataset

When the loading procedure is completed as shown in 48, the hierarchies for the quasi-attributes that will be anonymized must be imported.

3.7.2 Loading Hierarchies

The user can load hierarchies in two ways as depicted in figure 49:

1. **Option 1:** Pressing the ”Load New Hierarchy” button
2. **Option 2:** Pressing ”Load from Local” in the left menu.

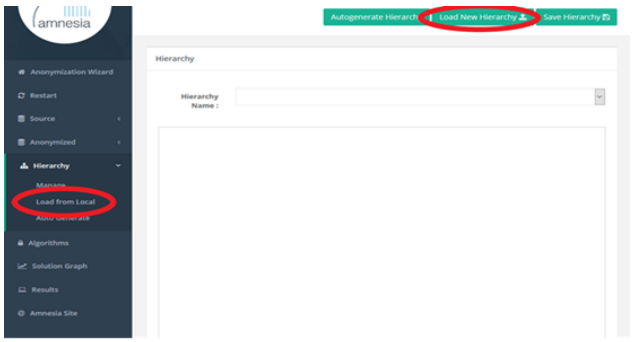

Figure 49: Hierarchy’s loading options

The hierarchies that already attached to the tool are presented as graphs, shown in figure 50.

Figure 51: Selection of a Attribute to Autogenerate an Hierarchy

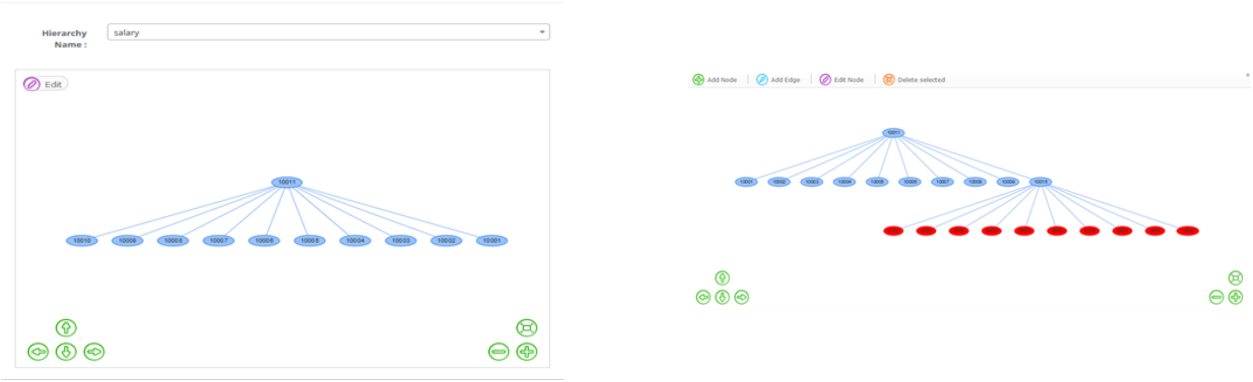

Figure 50: Amnesia Hierarchy Visualization

The user is capable to automatically create a hierarchy according to the dataset that he/she uploaded by clicking the ”Autogenerate hierarchy” button. In order to automatically create a hierarchy, the dataset must have been imported into the tool. Then the user should select the attribute from which the new hierarchy will be created and will be its leaves. Furthermore, the user is obliged to select the type of hierarchy (distinct hierarchy (distinct), range hierarchy (range)), but for string data type, there are grouping options (Group Based) and mask (Masking Based). After that, the user should give more information about the desired hierarchy, such as fanout, name, a sort method, etc., as illustrated in images 51, 52 below.

Finally, Amnesia allows the user to edit,delete or add a node in the hierarchy tree.

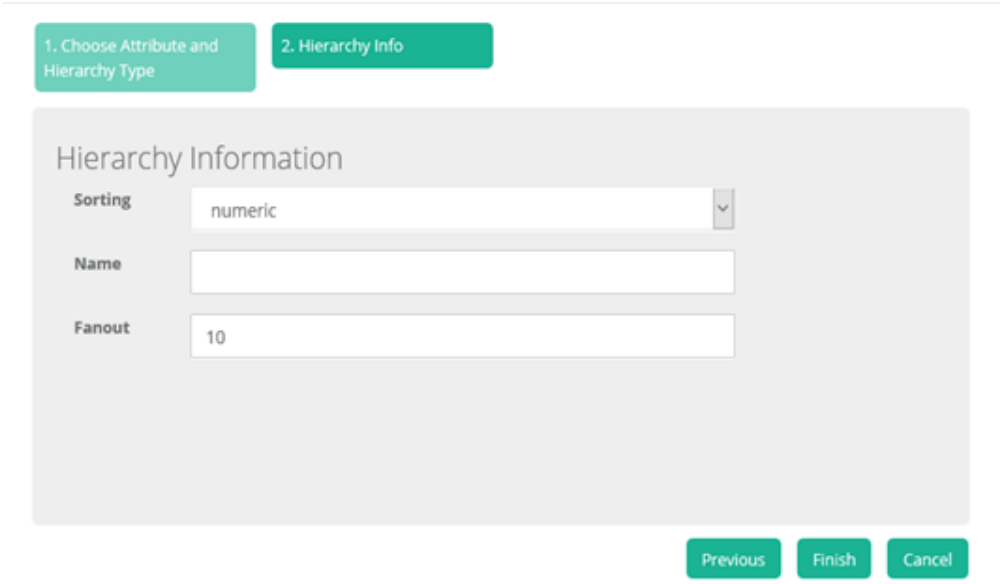

Figure 52: Additional Information about Hierarchy's Autogeneration

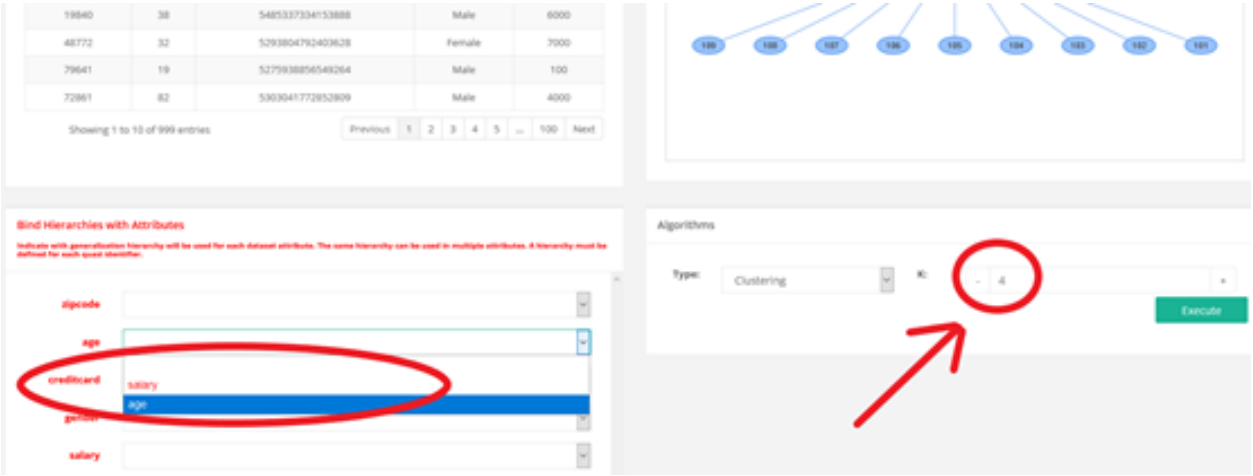

Figure 53: Bind Hierarchies and set the value of k

3.7.3 Algorithm Execution

After the data set and the hierarchies for the quasi-attributes have been entered, in order for the algorithm to be executed, the correlation of these hierarchies with the columns must be done as well as the value of k must be determined as indicated in the image 53 below.

The anonymous dataset is presented to the user next to the original when the algorithm is executed. As can be seen from the images 54,55, the value 58 has been anonymized to the value 105 in the first record, but in the fourth record, the value has remained the same as the original dataset proved the local recording anonymization approach.

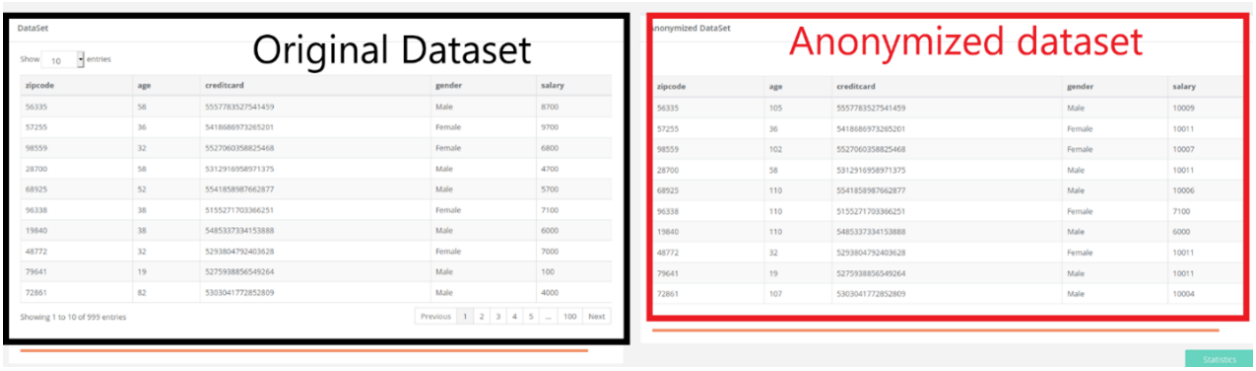

Figure 54: Anonymized and Original datasets

Figure 55: Local Recording

4. Conclusions & Future Work

4.1 Summary

The present age is characterized as the information age. Everyday life of the citizens is flooded with a huge amount of data through the use of computers and smartphones. Communication takes place almost exclusively digitally as well as the entertainment. While mainly in recent years, the work can be carried out online from home without a physical presence in the workplace. All these are accomplished by transferring and exchanging data via the internet.

In addition, the development of social networks as well as the rapid digitization of government services and documents make personal data vulnerable on the internet. Also, the huge data volume has led to the rapid growth of statistical science and research. However, there is the issue of personal data privacy and their secure processing without the leakage of individuals' personal information.

In this thesis we dealt with the problem of secure data processing by analyzing all anonymization techniques by presenting the relevant literature and examples in the main anonymization models. Moreover, we developed an anonymization algorithm for big data by storing the data on the hard disk via SQLite database and loading essential parts of the dataset into the main memory so as not to burden it. At the same time, a clustering method was

applied to satisfy the k-anonymity privacy model with local recording in the data to produce an anonymization dataset with less information loss. However, various problems and weaknesses of the algorithm were analyzed and solved. Besides, various issues and weaknesses of the algorithm were analyzed and solved. Additionally, the algorithm was evaluated through a one million records dataset in terms of execution time and information loss. A use case scenario of the algorithm was presented through the Amnesia anonymization tool for which the specific algorithm was developed.

4.2 Extensions

Our algorithm is quite satisfactory in terms of big data management and can pique the interest of many users for Amnesia as big data is now an integral part of the research. Despite the analyzed optimizations, the disk-based clustering algorithm can be further improved, especially in terms of time.

An improvement that will be made soon is to consider the number of records that exist in each cluster in the distance metric. Thus, there will be more data dispersion into the clusters so that the records do not converge in certain clusters reducing the splits and the unacceptable clusters with a number of records smaller than k, making the algorithm faster.

A simple implementation would be to deposit the number of cluster records as a weight on the already existing distance metric so the distance will be substantially longer, and one record will prefer the cluster which is essentially less similar but with fewer records so that the distribution of the record becomes more uniform.

Of course, the information loss will certainly be affected, but probably much less since there will no longer be many ”small” clusters and there will be no need to redistribute these records into ”big” clusters, this hypothesis will be revealed in practice.

An essential extension would be applying a differentially private mechanism for possible improvement of the algorithm and extension of the algorithm’s philosophy combining generalization and noise addition. As mentioned above, the concept of differential privacy is a very innovative and up-and-coming anonymization technique. In the existing algorithm, in order to avoid splits and redistributions of unacceptable clusters’ records, we could use a Laplace mechanism to add noise to the number of records with such a sensitivity to ensure that the number of each cluster would be in the range $[\mathrm{k} - (2\mathrm{k} - 1)]$. As a result, the algorithm will not need to split and redistribute the registrations of unacceptable clusters, which are quite time-consuming processes. Another consideration is the simple application of a differentially private mechanism when the k-anonymity through the existing algorithm has been completed and simply adding noise to the number of all combinations of quasi-identifiers. The latter idea may offer the combination of k-anonymity with differential privacy but does not provide a substantial improvement to our algorithm.

However, differential privacy is now an important procedure in order to achieve data privacy, and many researchers and private entities use it in their data. Thus it is imperative

need to apply differential privacy to this algorithm in order to reduce data corruption and maintain important statistics.

List of Figures

List of Tables

References

[1] L. Sweeney, "k-anonymity: A model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, vol. 10, pp. 557–570, 2002.

[2] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," 08 1998.

[3] G. Philippe, "Revisiting the uniqueness of simple demographics in the us population," 01 2006, pp. 77–80.

[4] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," 06 2008, pp. 111–125.

[5] ——, "How to break anonymity of the netflix prize dataset. arxiv cs/0610105," 01 2006.

[6] P. Ohm, "Broken promises of privacy: Responding to the surprising failure of anonymization," *UCLA Law Review*, vol. 57, 08 2009.

[7] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Trans. Knowl. Data Eng.*, vol. 13, pp. 1010–1027, 2001.

[8] C. Dwork, "Differential privacy," in *ICALP*, 2006.

[9] M. Hilbert and P. López, "The world's technological capacity to store, communicate, and compute information," *Science (New York, N.Y.)*, vol. 332, pp. 60–5, 02 2011.

[10] T. Breur, "Statistical power analysis and the contemporary "crisis" in social sciences," *Journal of Marketing Analytics*, vol. 4, 11 2016.

[11] D. Boyd and K. Crawford, "Six provocations for big data," 01 2017.

[12] J.-W. Byun, A. Kamra, E. Bertino, and N. Li, "Efficient k-anonymization using clustering techniques," vol. 4443, 02 2007, pp. 188–200.

[13] J. Li, R. Wong, A. Fu, and J. Pei, "Achieving k-anonymity by clustering in attribute hierarchical structures," 09 2006, pp. 405–416.

[14] J. Cao, P. Karras, C. Raïssi, and K.-L. Tan, "rho-uncertainty: Inference-proof transaction anonymization," *Proc. VLDB Endow.*, vol. 3, pp. 1033–1044, 2010.

[15] M. Terrovitis, N. Mamoulis, and P. Kalnis, "Privacypreserving anonymization of set-valued data," *PVLDB*, vol. 1, pp. 115–125, 08 2008.

[16] A. Pfitzmann and M. Hansen, "Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management–a consolidated proposal for terminology," *Version v0*, vol. 31, 01 2007.

[17] I. Neamatullah, M. Douglass, L.-w. Lehman, A. Reisner, M. Villarroel, W. Long, P. Szolovits, G. Moody, R. Mark, and G. Clifford, "Automated de-identification of free-text medical records," *BMC medical informatics and decision making*, vol. 8, p. 32, 07 2008.

[18] J. Celko, *Hierarchies in Data Modeling*, 12 2012, pp. 195–210.

[19] P. Samarati, "Protecting respondents' identities in microdata release." *Knowledge and Data Engineering, IEEE Transactions on*, vol. 13, pp. 1010–1027, 12 2001.

[20] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain k-anonymity." vol. 2005, 01 2005, pp. 49–60.

[21] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. Fu, "Utility-based anonymization using local recoding," vol. 2006, 01 2006, pp. 785–790.

[22] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "Fast data anonymization with low information loss." 01 2007, pp. 758–769.

[23] T. Iwuchukwu and J. Naughton, "K-anonymization as spatial indexing: Toward scalable and incremental anonymization," *VLDB 2007: Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 746–757, 01 2007.

[24] F. Kohlmayer, F. Prasser, C. Eckert, A. Kemper, and K. Kuhn, "Flash: Efficient, stable and optimal k-anonymity," 09 2012, pp. 708–717.

[25] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," vol. 2006, 05 2006, pp. 25 – 25.

[26] L. Sweeney, "Datafly: A system for providing anonymity in medical data." 01 1997, pp. 356–381.

[27] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," vol. 1, 01 2006, p. 24.

[28] J. Domingo-Ferrer and V. Torra, "A critique of k-anonymity and some of its enhancements," 03 2008, pp. 990–993.

[29] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," vol. 2, 05 2007, pp. 106 – 115.

[30] Y. Rubner, C. Tomasi, and L. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, pp. 99–121, 01 2000.

[31] R. Ahuja, T. Magnanti, and J. Orlin, "Networks flows: Theory, algorithms, and applications. prentice-hall, inc., upper saddle river," *New Jersey*, 01 1993.

[32] X. Xiao and Y. Tao, "Anatomy: Privacy and correlation preserving publication," 01 2006.

[33] ——, "M-invariance: Towards privacy preserving re-publication of dynamic datasets," 01 2007, pp. 689–700.

[34] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li, "Secure anonymization for incremental datasets," 01 2006, pp. 48–63.

[35] M. Nergiz, M. Atzori, and C. Clifton, "Hiding the presence of individuals from shared databases," 01 2007, pp. 665–676.

[36] M. Nergiz and C. Clifton, “δ-presence without complete world knowledge,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, pp. 868 – 883, 07 2010.

[37] O. Gkountouna, S. Angeli, A. Zigomitros, M. Terrovitis, and Y. Vassiliou, “km-anonymity for continuous data using dynamic hierarchies,” 09 2014, pp. 156–169.

[38] S. Warner, “Randomized response: A survey technique for eliminating evasive answer bias,” *Journal of the American Statistical Association*, vol. 60, pp. 63–6, 04 1965.

[39] H. Giggins and L. Brankovic, “Vicus: a noise addition technique for categorical data,” vol. 134, 12 2012, pp. 139–148.

[40] T. Fares, A. Khalil, and B. Mohamed, “Privacy preservation in data mining using additive noise.” 01 2008, pp. 50–55.

[41] D. Li, X. He, L. Cao, and H. Chen, “Permutation anonymization,” *Journal of Intelligent Information Systems*, vol. 47, 08 2015.

[42] T. Theeramunkong, C. Nattee, P. Adeodato, N. Chawla, P. Christen, P. Lenca, J. Poon, and G. Williams, “New frontiers in applied data mining,” 01 2010.

[43] J. Domingo-Ferrer and V. Torra, “A critique of k-anonymity and some of its enhancements,” 03 2008, pp. 990–993.

[44] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” vol. Vol. 3876, 01 2006, pp. 265–284.

[45] A. Wood, M. Altman, A. Bembenek, M. Bun, M. Gaboardi, J. Honaker, K. Nissim, D. O’Brien, T. Steinke, and S. Vadhan, “Differential privacy: A primer for a non-technical audience,” 01 2018.

[46] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” vol. 9, pp. 211–407, 01 2014.

[47] C. Dwork, “Differential privacy: A survey of results,” vol. 4978, 04 2008, pp. 1–19.

[48] J. Hsu, M. Gaboardi, A. Haeberlen, S. Khanna, A. Narayan, B. Pierce, and A. Roth, “Differential privacy: An economic method for choosing epsilon,” vol. 2014, 07 2014.

[49] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” vol. 4004, 05 2006, pp. 486–503.

[50] F. McSherry, “Privacy integrated queries: An extensible platform for privacy-preserving data analysis,” *Commun. ACM*, vol. 53, pp. 89–97, 09 2010.

[51] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, 01 2005.

[52] F. McSherry and K. Talwar, “Mechanism design via differential privacy,” 11 2007, pp. 94–103.

[53] A. Roth and T. Roughgarden, “Interactive privacy via the median mechanism,” 11 2009.

[54] V. Vapnik and A. Chervonenkis, “On the uniform convergence of relative frequencies of events to their probabilities,” *Theoretical Probabibility and its Applicactions*, vol. 17, pp. 264–280, 01 1971.

[55] J. Domingo-Ferrer, “Big data anonymization requirements vs privacy models,” 01 2018, pp. 305–312.

[56] V. Estivill-Castro, “Why so many clustering algorithms – a position paper,” *ACM SIGKDD Explorations Newsletter*, vol. 4, 06 2002.

[57] G. Loukides and J. Shao, “Capturing data usefulness and privacy protection in k-anonymisation,” 01 2007, pp. 370–374.

[58] C.-C. Chiu and C.-Y. Tsai, “A k-anonymity clustering method for effective data privacy preservation,” vol. 4632, 01 2007, pp. 89–99.

[59] H. Zhu and X. Ye, *Achieving k-Anonymity Via a Density-Based Clustering Method*, 06 2007, vol. 4505, pp. 745–752.

[60] J.-L. Lin and M. Cheng, “An efficient clustering method for k-anonymization,” 03 2008, pp. 46–50.

[61] W. Zheng, Z. Wang, T. Lv, Y. Ma, and C. Jia, “K-anonymity algorithm based on improved clustering,” 11 2018, pp. 462–476.

[62] J.-L. Lin, M. Cheng, C.-W. Li, and K.-C. Hsieh, “A hybrid method for k-anonymization,” 12 2008, pp. 385–390.

[63] R. Bayer and E. McCreight, “Organization and maintenance of large ordered indices,” *Acta Informatica - ACTA*, 01 1972.

[64] M. Crosas, “The dataverse network®: An open-source application for sharing, discovering and preserving data,” *D-Lib Magazine*, vol. 17, 01 2011.