# CHAPTER 3 – MULTI-DIMENTIONAL SENSITIVITY-BASED ANONYMIZATION METHOD

Data analytics and their utilization in big data environments witness a rapid growth in the past few years. Several undesirable side-effects have appeared, in relates to data disclosure and privacy violations risks. This trend imposes finding privacy methods with a scale-up ability to cope with the big data growth. Data anonymization is one of the pioneer privacy solutions that can minimize such risks. However, the current anonymization solutions suffer from poor performance and high loss of gained information in the big data environment. This Chapter introduces a novel privacy method named as Multi-Dimensional Sensitivity-Based Anonymization. The method resolves the performance and anonymization loss concern and provides a role-based anonymization control. Various privacy methods were proposed to anonymize data before exposing sensitive information on the cloud. The contemporary anonymization methods do not take the big data specifications into considerations.

Big Data analytics is where advanced analytic techniques operate on big datasets [1]. Hence, analytics is the main concern in big data, and it may be exploited by data miners to breach privacy [2]. In the past few years, several methods that address the data leakage concerns have been proposed for traditional data [3, 4]. The proposed methods provide remedies for variant types of attacks against data analytics process. Side attack is considered to be one of the most critical attacks [5]. Side attack method was explained in Chapter 2. This attack is prevalent in medical data, where the attacker owns partial information about the patient. The attacker aims to find the hidden sensitive information by logically linking between his/her data and the targeted data [6].

The previous Chapter has addressed both traditional and big data possible attacks and solutions. However, new methods did not consider big data specifications and behaviour. The proposed anonymization methods are inadequate for big data operation. This is correct even for the recently proposed anonymization methods. Recent methods, such as Two-Phase Multi-Dimensional Top down Specialization method (Two-Phase MDTDS) [7] , MapReduce TDS (MRTDS), or MapReduce BUG (MRBUG), do not mimic parallel processes and operations [8] . The mentioned methods can operate efficiently in traditional data. However, big data specifications and operations concepts are different. Big data operates in a parallel distributed environment, where performance and scalability are a major concern.

There is a need for an anonymization method that is able to operate in parallel. This imposes a need for changing the core structure of anonymization technique. Before proposing any anonymization method for big data, some specifications should considered. Developers need to distinguish the disparity between big data and traditional data. . With big data, anonymization process should be able to reduce the computation costs, prevent high information loss and increase security. Also, the anonymization process should be provided with a granular access control method. Hence, any big data anonymization developer should pay attention to the following specifications: equivalency increase, information gain, parallel algorithm, and gradual access. These specifications will be discussed in the next sections.

## Requirements for Big Data Anonymization Method

### Equivalency Increase

The equivalency increase is a general specification that must be considered on proposing any k-anonymity method for big data. This can be defined as;

**Lemma:** *In data records, the percentage of equivalent records proportioned extrusive with the increasing number of records. The rising number of records can help the least common attributes to gain the equivalency.*

This is true for most attributes. Few attributes are excluded, as a reason for their solitary nature like; emails, usernames, phone or fax numbers, and primary keys. To prove Lemma mathematically; let us denote the number of occurrences for any Q-ID value by *n*. Therefore, the Q-ID probability is calculated as; P(*qid*)=1/*n*. If the number of Q-IDs is *m*, then the number of occurrences of any Q-ID record is calculated by the factor of the Q-IDs;

(3.1)

Where P[QID] is the probability of each Q-ID.

For each record of Q-IDs, the number occurrences is given by. If we assume that each combination of values appears only once; then we need at least n records to gain one-time occurrence. Also, we need *k n* records to gain the *k-anonymity* for each combination of values. Referring to *k-anonymity*, the equivalency q is defined as the total number of equivalent records, where q ≥ k for each occurrence. For instance, if *k* =5, then each distinguished record must appear five times in N before gaining the *k-anonymity*. In the real data, the number of Q-IDs combinations appearances are usually less than n. Let us call the number of actual combinations appear is n̄. Based on our assumption of one time appearance for each combination; we can calculate the minimum value of N as:

(3.2)

Where n̄ denotes the number of actual combinations that appear in N, n̄ ⊂ n

Equation 2 assumes that each record has an equal number of appearances to the other records, which concludes that q=*Nmin*. However, in the real data, this is not a typical case. Thus, some records appear less frequently than the others, which makes some records reach the equivalency, while others fail. However, Equation 2 describes only one scenario. Nevertheless, any situation should consider the variable n̄. The probability value of variable n̄ remains between stability and increase, and it never decreases. In reality, the value of n̄ usually increase, while the stable scenario is less common. Besides, the equivalency q is proportioned extrusive with N, and can be described as q ≤ N, so q α N. This can be presented by the increase percentage of equivalency Q = q / N.

The positive relationship q ≤ N can be proven experimentally. Three various experiments were conducted by using the adult database from the UCI Machine Learning Repository [9]. The database describes the Age of adults, their Occupation, Marital status, Education, Social status, Position, Sex, Hours worked per week, Race, County, Native country, and salary. We assigned four Q-ID attributes; Age, Education, Salary, and Sex. The experiments are conducted using MatLab simulator [10], by choosing three groups of N records small, medium and large.

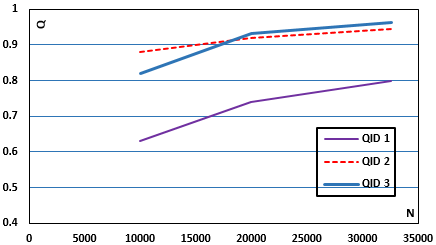
During the experiments it is assumed that k = 10, and the Q-IDs probabilities are calculated as; P[age] = P [1-100] =0.01, P[education] = P [Y5-6, Y7-8, Y9, Y10, Y11, Y12, HS-grade, Some-college] = 0.125, P[sex] = P [Male, Female] = 0.5, and P[S] = P [<=50K , >50K] = 0.5. Hence, the maximum number of combinations is calculated as.

In the first experiment; the total number of records was N =10,000 records. The number of the actual appearing combinations in 10,000 records was n̄ = 1741, which presents around 50% of the probable appearances. The number of equivalent records is q=6272, which presents around Q=60% of the total number of records. In the second experiment, the number of records was increased up to N= 20,000. The number of actual appearing combinations n̄ = 2196, which presents around 69% of the probable appearances. The number of equivalent records is q=14828, which presents around Q=75% of the total number of records. In the third experiment; the number of records was further increased up to N=32,561. The number of actual appearing combinations is n̄ = 2498, which presents around 78% of the probable appearances. The number of equivalent records is q=26846, which presents around Q=82% of the total number of records.

Similar steps were applied on three groups of randomly picked Q-ID’s in the same dataset. The three groups are shown in Table 3.1. The Q value was calculated for each group. All groups showed an increase in both of the equivalency percentage Q, as shown in Figure 3.1, and the actual appearing combination n̄, as shown in Figure 3.2.

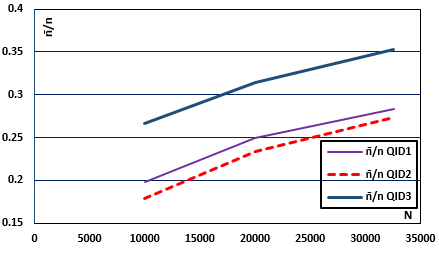
*Table 3.1- Four Q-ID groups chosen randomly from Adult dataset*

|  |  |  |
| --- | --- | --- |
| **QID Group1** | **QID Group2** | **QID Group3** |
| Age | Marital status | Position |
| Education | Education | County |
| Sex | Social status | Country |
| Salary | Race |  |



*Figure 3.1- Proportionality of equivalent records with increased record numbers.*

.



*Figure 3.2- The effect of increasing the number of actual combinations with increased number of records*

Both diagrams prove that data growth supports data equivalency in big datasets. For this reason, we need to find a proper algorithm that is able to process a large data size without a need for spitting data into small chunks with a large data processing capacity.

### Information Gain and Security

Current anonymization methods are mostly adaptations of approaches that were designed for traditional data [11]. The traditional data concludes a limited number of records, therefore, information gain is an essential matter. However, big data equivalency increases extrusive with the data size increase, which concludes a larger group of equivalent records. Thus, the information gain concern may not be the prime factor in anonymization process. Logically, more equivalent data cause less anonymized records. Also, all indications point out security concerns in big data. The increased number of users’ access and the massive quantity of personal information in datasets may increase the re-identification probability. Hence, the trade-off between security and information gain may result in favour of security. Therefore, anonymization algorithms should pay more attention to security concerns. Algorithms with several times iterations, to find out the best Q-ID anonymization, the best k value, or the best cut and interval is inadequate in big data. The multiple iteration and scanning is an expensive computation cost. Moreover, the high level of accuracy and the small value of information gain increase does not really affect the statistical and analytical results.

One of the major distinguishing features in big data is the multi-dimensionality [12]. This leads to a large number of Quasi Identifiers. The previously proposed methods suggested limited number of Q-IDs, which may reach up to seven or eight. However, when data is multi-dimensional, then more expected Q-IDs will join the data records. The large number of Q-IDs leads to a higher security threat. Re-identification becomes easier with the increased number of auxiliaries and data identifiers. More personal information revealed will definitely facilitate the side link attack or the background knowledge attack. The attack possibility becomes even higher with the new technology evolution of social media, smart phones and cloud services. Users are now able to find out most personal details over the internet. Hence, more security vigilance is essential.

The above facts do not mean a total ignorance for the information gained by the anonymization method, but reducing the processing of finding the optimal anonymization values instead. The optimal values usually lead to the lowest possible loss of information after anonymization. In big data, applying such complicated processes may not affect the final results of statistical output. The small statistical values can be even ignored, since the statistical results follow the principle of estimation prospects. This gives data miners a flexibility of approximating and rounding some numbers to few decimal places [13]. Therefore, pre-calculating the k value, and pre-determining the attributes needed to be anonymized is an advantage. Generally, this non-accuracy will not dramatically affect the data analytics results.

Moreover, big data is beneficial when it is public, this means many organizations from different fields need to access this data for multiple purposes [14]. They all analyze, mine, and output statistical results. This fact emphasizes the security manifestation side in big data. Multiple users from various organizations may need to access data with different levels of access. Users cannot be given similar security level, therefore, granularity is required as a part of security procedures. This will be further explained later.

### Parallel Algorithm

A parallel distributed environment handles big data. The multi-task processes should be considered in any anonymization method used for big data. This can be implemented by splitting tasks into sub-tasks, and distributing them among multi-computers to cope with the massive data volume [15]. Parallel programming model is provided by the processing framework such as; MapReduce or Spark. The framework provides parallel operations for reading/writing data from/to disks, storing data in memory, and processing data in parallel. These three procedures can be completed by the big data frameworks. Parallel programming model is classified into two areas; process interaction and problem decomposition. The process interaction is related to managing shared memories to accommodate the processed data, passing messages between nodes, and dividing large tasks into smaller subtasks [16].

Many obstacles face parallel programming models, such as; network speed and massage passing between nodes. This causes a bottleneck of sequential operations in shuffling phases. Also, finding a general valuable programming model is difficult. Any parallel model is judged on its generality, since some models may perform efficiently in specific data types and environments [17]. For instance, Flink and Storm frameworks perform well in multimedia and data stream, while Hadoop Distributed File System reforms well on large data files. Parallel programing adopts the implicit and explicit parallelism concepts, whereas implicit parallelization converts the program codes from sequential into parallel automatically. The explicit programming is difficult and requires high programming skills, to enable several nodes to communicate across the network. For this reason, current parallel models for big data are partially implicit [18].

In big data, the processing framework, such as MapReduce, handles the majority of the implicit parallelization. However, some parts of the parallelism relies on the application programmer, who should be aware of the framework core structure. The programmer needs to know the optimal code that may result the best performance. This can be implemented by several trials of running the program, to arrange the commands of filtering, grouping, replacing and others in the best sequence. In big data anonymization, the programming commands control the dataflow operands without changing the program’s state. The programmer no longer specifies the detailed sequence of instructions in execution orders, but rather the general operations to be applied to the dataflow. Eventually, the programmer has a small window of managing the operations sequence, hence, the anonymization framework should consider this level of programming.

### Gradual Access

Any proposed big data framework should consider the large number of demands on analytics. The more data growth, will lead to a higher number of analyser’s requests. The large data size is mostly generated by a large number of participants, hence, they contain a great percentage of personal information. As a result, analysing this massive data will enhance the future plans and strategies in marketing, development, and decision making [19]. Because big data nature is public, and various analysers may wish to participate in data analytics, then there is need for considering the security term as the first priority. One of the main security components is the fine-grained access control for users. This implies a gradual access as per user’s right and privileges.

The gradual access should be provided through the anonymization control, by increasing or decreasing the anonymity level. Users with high privileges should be able to gain higher information levels. Therefore, the statistical analytics results should be close similar to the original data analytics results. On the contrast, users with low privileges should gain lower information levels. This data masking increase or decrease keeps data in the hands of the authorized users and organizations. The access control should be able to hide some data, for certain users. Hence, the access control can view or hide only. The view is masked with levels of data anonymization. The granular access should not have a large impact on analytics performance.

## Multi-Dimensional Sensitivity-Based Anonymization Method Concept

Multi-Dimensional Sensitivity-Based Anonymization (MDSBA) method is developed to fulfil the previous four specifications. These are equivalency increase, information gain, parallelism, and user access disparity. MDSBA adapts a multi-dimension technique for performing a high level of computation in big data. The MDSBA method mandates to define the privacy method and masking pattern for each access level. The novel anonymization method applies the Bottom-Up Generalization (BUG) in *k-anonymity* that can cope with the big data frameworks. The method does not only parallelize data for big data frameworks, but also reduces the overhead computation of data iteration. This is accomplished by providing pre-calculated *k-anonymity* parameters and pre-determined attributes for anonymization. The MDSBA also supports the anonymization-based access control. This imposes a gradual anonymization based on user’s access level. MDSBA mimics role-base access control by providing a granular security access for multi-user levels.

MDSBA consists of four main concepts, these are the definition of datasets, the probability concepts, the grouping, and the sensitivity value. Following this introductory, let us delve in MDSBA core concepts.

### Quasi Identifiers and Classes

The MDSBA general definition is similar to the previously proposed *k-anonymity* methods. Data owners choose some private attributes as Quasi Identifiers (Q-ID). The chosen Q-IDs contain personal attributes that may facilitate adversary’s tasks of re-identifying some records. More importantly, they may unveil some sensitive attributes, known by classes. For this reason, some of the chosen Q-IDs are anonymized, if they do not fulfill the *k-anonymity* rule. This imposes three different types of data attributes; Q-ID attributes, sensitive attributes (classes), and ordinary attribute. This can be defied as a table T consists of *m* set of attributes, where each attribute can be either Q-ID, class C, or an ordinary attribute *attr*. Hence T is a combination of T= {QID, C, *attr*}.

The chosen Q-IDs and C are pre-determined by data owners. There is no clear instructions on picking up these identifiers. MDSBA provides the best practice advices for determining Q-ID’s. This will be further explained in chapter six. Data owners setup the initial values of their own, including the Q-ID attributes decision. Data owners are the only users who are permitted to access the original copy of data exclusively. The anonymization level increases gradually with the ownership increase.

### Probability Concept and Anonymization Masking

MDSBA adopts Q-ID probability concept on grouping the Q-IDs and on masking process. The anonymization is conducted by applying one of the three masking tools, taxonomy tree, interval, or suppression. The probability is pre-calculated for each Q-ID related to the possible values that may appear in all data records. For instance the attribute probability of a person’s age is P(age)=1/100, if we suppose that the human age range is between 1 to 100. The Q-ID probability should be pre-calculated to accelerate the grouping in the anonymization process. The number of Q-ID values indicates how often the arbitrary appearance of each value. If the values are unique or non-repetitive, then the attribute cannot be an elected as a Q–ID attribute. Thus, the values must be finite and repetitive.

* **Probability on Grouping Q-IDs:**

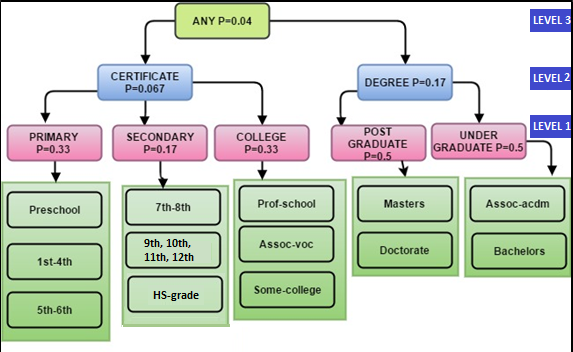
As mentioned before, the Q-ID probability is essential in grouping records. The grouping, in MDSBA, is conducted on a gradual basis. This allows grouping for all Q-IDs first, and then another grouping for all but one, and then for all but two and so on. In general, the lower probability Q-ID values may appear less frequently in data attributes. This imposes that data records with high Q-ID probabilities may return less data equivalency, and as a result higher anonymization output. For this reason, MDSBA meant to reduce the anonymization output by grouping the Q-ID attributes with the highest probabilities. MDSBA, divides the grouping task into multiple tasks, by grouping all Q-ID attributes in the first stage, so this can filter out the fully-equivalent records. In the second stage, the lowest probability Q-ID is excluded from the second grouping process. In the third stage, the lowest two Q-ID attributes are excluded from the third grouping process. The final stage groups only one Q-ID attribute, which is the one with the highest probability.

* **Probability On Masking:**

The probability is also used on masking Q-ID attributes. This probability is derived from the taxonomy tree concept. The taxonomy tree T is propagated from the parent node w to a number of leaf nodes ν, so each parent node’s probability is. Figure 3.3 illustrates probabilities for each parent node in the education tree. The two major anonymization tools are presented by taxonomy tree and intervals. This depends on the data type, since some data can be generalized by taxonomy trees, such as (Education). In contrast, other data types are numerical, and can best fit interval masking. The intervals, also, can be presented by propability values. If a number n was presented in an interval of a minimum value *Vmin*, and a maximum value *Vmax*, then the propability of obtaining that number within the interval range is . For example, for an interval of [15 – 25[, the probability is (P=1/10=0.1). This probability concept supports the fine-grained access for multiple users.

#### Interval and Taxonomy Tree Masking

The probability, in MDSBA, is the core concept of giving a gradual level of information gained from data. Users with higher privileges are given more accurate data due to their high trust level. The higher trust they have leads to a higher information gain. The trust level is determined by several factors that will be discussed later. To illustrate the garrulity in information gain, let us consider Figure 3.3 again. The higher trusted users may receive data masked with level one taxonomy tree. This imposes more accurate information about people’s education level. If a user was given an access level with a maximum probability of P(user)=0.17, then data sample appear in Table 3.2-A will be anonymized as per EDU taxonomy tree. The anonymization results show that one value was masked by level 1, while the rest of the values are masked by level 2. This is because the probability of the secondary node in the tree is 0.17, and the certificate probability is 0.067 < 0.17. In another example, if a user’s maximum probability P(user)=0.5, then the given masking will reveal more information, as shown in Table 3.2-C. Tables 3.2-B and 3.2-C are examples of access granularity as per user’s access level.

****

*Figure 3.3- Taxonomy tree for EDU in Adult data.*

*Table 3.2-A. Original Data Table 3.2-B. Masking with P ≤0.13 Table 3.2-C. Masking with P ≤0.5*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EDU** |  | **EDU** |  | **EDU** |
| Master | Degree |  | Post Graduate |
| Bachelor | Degree |  | Under Graduate |
| 9th | Secondary |  | Secondary |
| Some-college | Certificate |  | College |
| 5th-6th | Certificate |  | Primary |

In the previous examples, users are given probability values to control their access level. However, we need a mechanism to assign this probability value. One of the suggested methods was the k-anonymity value. In order to control the granularity, let us define an instant of k-anonymity value that is given for each user. Suppose that the k-anonymity value is k, while another user was given a value of k1, where k1 < k. in this case the smaller values of k impose less anonymization impact. Therefore, the user with k1 value will gain more information. The probability value**,** given to each user, can be better controlled by given multiple values of k-anonymity as per user’s access level. Thus, users with higher values of k may gain less information than users with lower values of k. This enhances the probability concept and provide better tools to control the access.

Referring the k-anonymity value and the granularity, let us define an instant of k, denoted by , and can be named as the ownership level. It is a number given to each user on accessing data for analytics. This number indicates the minimum number of equivalent Q-ID records to avoid anonymization. Larger value of implies a higher level of anonymization. The probability value for each user can be given a name of Sensitivity level, and is denoted by ψ. The sensitivity level ψ is calculated by giving the ownership level for users. Data owners determine the Q-ID attributes, and the value of *k*, then, the level of sensitivity is determined by MDSBA equations as per given for each user. The Sensitivity Level of the class attribute C is denoted by, and the ownership level of a user is indicated by  = k-i, where i= {k-1,… , 2} and 2≤ ≤ *k*.

#### Suppression Masking

Anonymizing data can be processed by applying three different types of masking, taxonomy tree, interval and suppression. The suppression imposes a full or partial data hiding, so zero or low information are gained for the suppressed data. MDSBA implements suppression in some cases, such as non-equivalent records after completing all grouping stages. This is usually applied during the final grouping stage, when grouping cannot find any semi-equivalent records. Moreover, suppression can be a proper solution for some data types. For instance, data of postcode, first name, last name, and contact numbers. These data better anonymized by applying special characters to replace the actual value. The common used character is the star (\*). Also, the suppression can be applied by using the word (any).

The probability can be applied with suppressing mask. For instance, if a person’s maximum probability is 0.04, then the name (Mark) can be masked by one letter and displayed as (Mar\*). The omitted letter’s probability P(letter)=1/26=0.04. Similar concept is applied on postcodes, for instance a postcode of (4514) manifests a value of 0.1 for each suppressed number. Hence, the previous postcode can be given by (45\*\*) for the same user. This concept is quite similar to intervals and taxonomy trees. However, some data types may accept either suppression, interval, or taxonomy. Therefore, data owner needs to determine the masking tool used in each data attribute. For example, the previous postcode can be anonymized by an interval of [4510 – 4535[. If compared between both masking results, then the interval may produce more accurate anonymization output. In the previous examples, the suppression process has omitted two numerical values from the postcode. The probability of the omitted numbers = 0.01, while the user is given a probability up to 0.04. However, it is not possible to omit one number of the postcode, because the probability of omitting one number is P=0.1, while the maximum given probability is P=0.04 On the other hand, the interval masking can accurately assign the exact probability. In the previous example, the user’s probability is 0.04, which is presented as an interval of 1/0.04 = 25. The postcode can start from 4510 with an interval of 25, so the maximum interval value is 4535. In case if the interval or taxonomy are more accurate, then it is better ignoring the suppression masking, or keeping suppression as a last resort.

In some cases, it is possible that user’s given probability is much lower than the probability of the taxonomy tree root. In such cases, the probability concept is ignored, and data is fully suppressed. For instance, if the maximum probability P(user)=0.01was given to a user. Referring to the previous EDU taxonomy tree, the root probability of P(Any)=0.04. Based on the probability concept, the EDU data cannot be anonymized by the value (Any), because P(user) < P(Any). However, (Any) is the highest level in the tree, and there is no any further generalization can be applied. Therefore, the masking algorithm generalizes any EDU value with (Any), if the user’s probability P(user) ≤ 0.04.

### Grouping Data Vertically and Horizontally

In order to parallelize massive size of data, and to support the anonymization granularity, MDSBA logically divides the Quasi Identifiers (Q-ID) vertically and horizontally. This technique supports several aspects in big data. In section 1.1, four different requirements were identified for any anonymization method. The grouping concept, in MDSBA, supports the anonymization requirements of equivalency, security, parallelization, and gradually. Vertical grouping divides the Q-IDs into small groups, so they can be an advantage for big data anonymization. When grouping data horizontally, more attributes can be chosen to be Q-IDs, which enhances the security level of the anonymized data. This technique splits large data into smaller data size to support the parallelism. Each group of data is sent to separate node, and processed in parallel. Also, vertical grouping enhances the access control method. Since users will be give the needed Q-ID groups only. The non-needed groups will be hidden, so users will be given the right amount of data only. Eventually, vertical grouping reduces he computation costs, so the computation process is conducted individually for each Q-ID.

Horizontal grouping enhances the information gained by splitting data logically instead of random split. This is further explained in chapter 5. Moreover, this grouping mimics the MapReduce framework by staging data from map to reduce. In horizontal grouping, data is grouped several times, and in each time, data becomes smaller. In the horizontal grouping, algorithms do not need to load a massive size of data with multiple iteration times, instead, data is dramatically reduced on each iteration time. This grouping method enhances the information gain, the equivalency increase, and the granularity. Let us further study each type of vertical and horizontal grouping.

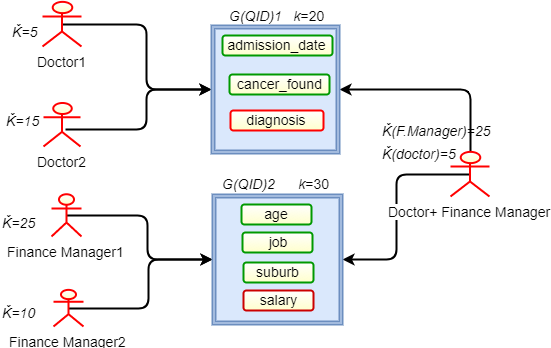
#### Grouping Data Vertically

MDSBA splits Q-IDs in to small groups of two to four Q-IDs. The groups are chosen based on the business roles. Personal information and related auxiliaries can be categorized based on user’s interests. In MDSBA, the aim is including all or most personal attributes in Q-IDs. This increases the total number of Q-IDs, so there should a way of dealing with this large number of Q-IDs. Referring to user’s various needs and to the increased number of Q-IDs, it is possible that each Q-ID group can be mapped to a business role. For example, Human Resource Administrator may focus on the patient’s address, age, and salary, while Radiologist may care about the cancer’s type, status, and size. The mapping between Q-ID groups and roles will be discussed in chapter 5.

We may describe the vertical definition by denoting n as a number of records, and m as a total number of attributes. The data owner defines the number of quasi-identifiers as *Q* in *m* attributes, so *Q* ≤ *m*. Each two to four Q-IDs are aggregated in a group G(QID), so the number of created groups, denoted by ϒ, is related to the total number of Q, and can be presented by , where . Each non-overlapped G(QID) group consists of Q-IDs and one class, which is usually mapped to one or more business roles R. Let us also denote U as a user, so the role-based anonymization control is presented as; {G 🡪 R}(many to many relationship), and {R 🡪 U}(many to many relationship). The vertical grouping principle is described by the following definition;

**Definition 3.1**: *a table T contains a Q number of Q-IDs, and a C number of classes. Q-IDs are grouped vertically by dividing Q and C attributes into* ϒ *groups. Each G(QID)* *group contains one class attribute and some two to four Q-ID attributes. In other words, 2 ≤ G(QID) ≤4, and C= 1.*

The vertical grouping divides Q attributes into small groups of two to four Q-IDs, with one class attribute. Each Q-ID group, G(QID), is mapped to a business role, and given a fixed value of k. In such a way, users are given authorization rights to access some Q-ID groups as per their given business role. Let us study the following example to illustrate the access control structure. If five users={user1, user2,…,user5} have requested to access the following Q-ID groups; G(QID)1={admission\_date, cancer\_found(yes/No), diagnosis(class)}, and G(QID)2={age, job, suburb, salary(class)}. G(QID)1 is mapped to Doctor, and given a value of k=20, and G(QID)2 is mapped to Finance Manager and given a value of k=30. Suppose that users are given the following roles {user1(Doctor1), user2(Doctor2), user3(F. Manager1), user4 (F.Manager2), user5(Doctor+F.Manager). Each one of these users will be given a value of to represent the ownership level. The values are shown in Figure 3.4. This figure illustrates the core base of access control management in MDSBA. The access granularity is presented by giving the G(QID) groups fixed k values, and giving users various values of . The k value determines the maximum optimal value for each group G(QID). Chapter 6 illustrates the reason for choosing the k value for each G(QID). Each user is given a value of , which determines the user’s access level. Users with higher access privileges obtain low values of , and vice versa. In Figure 3.4, Doctor1 gains more information than Doctor2 on accessing G)QID)1, while both doctors are not permitted to access G(QID)2. Similarly, Finance Manager 2 gains more information than Finance Manager 1 on accessing G(QID)2, while both managers are not permitted to access G(QID)1. The figure also shows tat it is possible to assign more than one business role to a specific user. In Figure 3.4 user 5 has the permission to access both groups, since he/she was assigned to both business roles. User 5 will gain high information from G(QID)1 and low information from G(QID)2.



*Figure 3.4- Access Control structure in MDSBA*

#### Grouping Data Horizontally

It is evident that records equivalency, in most datasets, increase in parallel with data growth [20]. Therefore, splitting data nominally may increase the data equivalency. Random data split ignores the distribution of equivalent records amongst tables. For instance, in Adult data, patient’s age, sex, and education attributes may appear similar in the first one thousand records, and then related records may appear again at the end of the table. Hence, the random appearance of similar records may reduce the number of equivalency on splitting data into small random chunks, which results in an increase of data masking.

The following four definitions describe the horizontal split of data by implementing the grouping method of equivalency in MDSBA. It was proven earlier that splitting data nominally will increase the equivalency ratio. Suppose a data set D, with a total number of *m* attributes, and *n* records. The data owner defines the number of quasi-identifiers as *Q* in *m* attributes, so *Q* ≤ *m*. Let a number of *q* attributes have equivalent records *k*, where *q* attributes are part of *Q*, so *q* ≤ *Q* ≤ *m*. Hence, the wholly or partly similar records are defined as follows:

**Definition 3.2:** *All D records are split based on the class C values. Each set of records that contains similar class value is aggregated in a Gi group, where i denotes the number of values appear in the class C. Every Gi group is further processed individually.*

**Definition 3.3:** *The fully-equivalent group (SG) contains some k equivalent records, in some Q attributes. For this group, there is no any anonymization process applied.*

**Definition 3.4**: *The Semi-equivalent group (SSG) contains some k equivalent records, in some q attributes, where 2≤ q ≤Q-1. The highest Q-IDs probability is usually chosen for q attributes equivalency. The anonymization is applied on the rest of non-equivalent Q-IDs.*

**Definition 3.5**: *The non-equivalent group (NG) contains a number k of equivalent records, in some q attributes, where the number of q=1.The highest Q-ID probability is usually chosen for q attribute. The anonymization is applied to the rest of non-equivalent Q-IDs.*

As explained in Definitions 3.4 and 3.5, the SSG and NG anonymization are applied on the lowest Q-ID probabilities. For instance, in Adult data the Q-IDs probabilities are; P[Age]= 0.01, P[Sex]= 0.5, P[Edu]= 0.08. Based on these values, the SSG anonymization will be applied on the Q-ID with the lowest probability, which is [Age]. So the semi-equivalency is measured by grouping Sex, and Edu attributes, while Age attribute is anonymized as per interval. The NG anonymization will be applied to all Q-IDs except the one with the highest probability. In our example, the anonymization will be applied to [Edu] and [Age], while records are grouped as per [Sex] equivalency.

MDSBA framework splits data recursively to create three dependent stages of MapReduce processes. Stage one aims to produce Gi groups for the whole dataset with one Pig command only, which filters each data record based on its class attribute, as described in Definition 3.2. The number of the produced Gi groups relies on the number of sensitive class values. For instance, Seer Cancer data contains four values of sensitive attribute (Cancer); these are {no positive histology, Positive histology, Positive microscopic confirm, and Positive laboratory test}. The four values create four groups of Gi ={G0,G1,G2,G3}. Each group is stored in a separate HDFS location. Stage two reprocesses G groups in parallel, to categorize data records between wholly or partly equivalent or between SG, and SSG. According to definitions 3.3 and 3.4, the SG groups denote the equivalent Q-ID group, while the SSG groups denote Semi-equivalent group. The SG groups are equivalent, so anonymization is skipped, and data is stored in an output directory. The third stage’s input is derived from the second stage’s output, so SSG groups are stored in an input directory waiting for stage three. The process starts by reading SSG groups and abstracts the non-equivalent groups, denoted by NG. Both of NG and SSG are anonymized by using User Defined Function Java program.

In Adult data example, Q-ID attributes are grouped by the three Q-IDs (Age, Sex, and Edu), so the number of equivalent records must be greater than or equal to. These equivalent records are stored in SG, while the equivalent records with a number smaller than are stored in SSG. In the second stage, the three attributes are grouped again by the largest probability values (Sex, and Edu). The equivalent records with a number greater than or equal to will be anonymized. The anonymization is applied on the attribute with the lowest probability value, which is Age. The equivalent records with a number smaller than are stored in NG, in order to be further grouped and anonymized.

### Mathematical Equations to Calculate the Sensitivity Level

As explained before, data is split into several groups or domains vertically and horizontally. In each Q-ID group, a different anonymization is applied depending on the user’s access level. The anonymization process is managed by the value of sensitivity level ψ, which increases or decreases the information gained from the data. User with a higher value of ψ gain more information, and vice versa. The sensitivity level is determined by two major factors: the sensitivity factor ω and the aging factor τ. In datasets, the value of represents the *k-anonymity,* and represents the ownership level. A large value of ownership level implies a weak ownership relation, so the weakest ownership relation is when. The integer values of and are such that. In other words, low values of correspond to reduced anonymity as a result of higher ownership relations. The sensitivity factor ω is calculated based on its maximum and the minimum probability values. The maximum probability value of ω is defined as:

(3.3)

The minimum value of ω is defined as the product of all Q-IDs probabilities, or:

(3.4)

Based on Equations 1 and 2, the value of ω can be found linearly between ωmin and ωmax, as shown in Equation 3:

(3.5)

Equation 3 presents a linear relation between ω andvalues. The equation lowers the value of ω when the user’s value is high, so that the data anonymity is higher. Users with less privileges are given higher, while the value of k is constant. We also propose that the lowest value of should be two to avoid unique re-identification. The anonymization is only applied on external organization’s access. External organizations are permitted to access an anonymized copy of the original data, while accessing the original data is exclusive to the data owners.

Equation 4 collates both terms of ω and τ to compute the sensitivity level ψ. The sensitivity level directly manages user’s access privileges. Higher sensitivity levels lead to upper access ranks, where less data masking and concealments are applied. Also, the sensitive nature of an object can be made lower as its related data set ages. The aging factor τ affects the sensitivity reversely, with considering the negative values of aging factor. Based on their owners’ decisions, old objects can be deemed to be less sensitive, and increase ψ as they age. In essence, two factors determine the sensitivity level of the data, the sensitivity factor ω, and the aging factor τ, as described Equation 4.

(3.6)

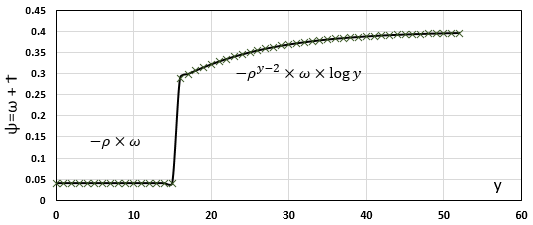
Equation 4 is used to calculate the sensitivity level ψ, by establishing the sensitivity level of an object for a given user access level. The masking process tends to find any number close or smaller than the sensitivity value. For instance, if ψ =0.5, then any value between 0-0.5 is acceptable. However, the values closer to ψ improve the information usefulness and the granularity precision. For instance, the age interval of [10 – 20[is derived from ψ =0.1, while the age interval of [10 – 12[is derived from ψ =0.5. We may notice the accuracy difference between both values of ψ, therefore, assigning the closer value to ψ is essential. The aging factor creates a perturbation to the sensitivity value. This factor becomes more significant as the data age becomes older than a particular age that we refer to as its obsolescence value. Equations 3 and 4 show that the lower sensitivity level requires a higher value of, which corresponds to lower information gain. In other words, the lower sensitivity levels, correspond to higher anonymization and masking levels of information.

#### Sensitivity Level and Time Factor

The aging factor τ depends on four different parameters, the object obsolescence value Ø, the aging participation percentage ρ, the object age y, and the sensitivity factor ω. The Ø value is defined as the critical age before the object sensitivity starts to degrade. It can be measured by units of hours, days, weeks, months or years, which totally depends on objects obsolescence speed. However, Ø cannot be given a value less than 2, so the value of one year, for example, can be replaced by 12 months instead. The aging participation percentage ρ is an approximation percentage chosen by data owners. It measures the aging factor participation in data objects. The aging factor τ can remain constant or decrease linearly. It remains unchanged when the age of the object y is less than its obsolescence value, that is it remains constant if y < Ø. On the other hand, if the object age is greater than or equal to its obsolescence value, that is y≥ Ø, then sensitivity level increases logarithmically, which in turn decreases anonymization level. These two cases manage the aging factor τ, as described by Equation 5.

(3.7)

Data owners may set ρ to 0% if their data objects are not affected by time and age. Also, the maximum value of ρ cannot exceed the 90% to avoid a nil value for ψ. It can be noted from Equation 5 that τ is always negative. In other words, the sensitivity level ψ increases with the age of the information and the passage of time. The newly created data objects result in a lower sensitivity level compared to the old data. In other words, if *y* <Ø then ψ < ω when the aging factor is incorporated. Equation 5 is derived based on our proposed ideas for incorporating the aging factor and sensitivity analysis for improving the anonymization process. It relies on a linear condition -ρ × ω and a semi-log component (-ρ(y-2) × ω × logy). The linear part produces a constant value of aging factor when the information has not reached its obsolescence value. The semi-log portion is derived from the plotted graph, shown in Figure 3-5. The figure illustrates the concept that incorporates the age of data objects. Older data is considered to have a logistic degradation with age and passage of time. The degradation starts swiftly before plateauing at the sensitivity factor value ω. To illustrate the time factor impact, let us consider the following example: a social media data has an aging participation value set at 0.9 with obsolescence value of 15 days and a sensitivity factor of 0.4 (ρ=0.9, ω=0.4, and Ø=15). Plotting its sensitivity diagram can be initiated by assuming a constant value of sensitivity factor multiplied by the aging participation percentage to find out the aging factor. Therefore, τ=-ρ × ω= -0.36, and ψ=0.04. Next, a logistic graph can describe the sensitivity level direct proportions the variable y increase. The sensitivity level increases dramatically between days 15 and 30. This sharp increase describes the object degradation importance, which will intentionally reduce the obscurity level on data anonymization.



*Figure 3.5- Plotted graph to derive Equation 5 for aging factor*

The following example illustrates the calculation steps of sensitivity value ψ. Consider an object with three Q-ID attributes, say student IQ test results, similar to those shown in Table 3.3. The data owner intends to anonymize the data with *k*=20, the obsolescence value Ø is set to 10, the aging participation ρ is 70%, and the age of the object y is13 years. Now, suppose a user was given an ownership value =10. Based on the given attributes, the sensitivity level ψ can be calculate using Equations 4 and 5. To find out the sensitivity factor ω, it can be noted that the values of ωmax=max (0.01, 0.005, 0.125) is 0.125 and ωmin=0.01 × 0.005 × 0.125 is 6.25 × 10-6. The value of ω as per Equation 3 is ω≈ 0.063. Based on Equation 5, the aging factor τ=-(0.7)11 × 0.063 × log (13) can be seen to be -0.00138. Therefore, the sensitivity level ψ = 0.063 – 0.00138 is found to be 0.062.

*Table 3.3- Three Q-ID attributes example*

|  |  |  |
| --- | --- | --- |
| **Q-id(attribute)** | **Q-id type** | **Probability** |
| Q-id0(IQ\_value) | Interval IQ\_value=[50-150] | P(Q-ID0)=1/(150-50)=0.01 |
| Q-id1  (Student\_country) | Taxonomy tree Student\_Country\_Level1 = {German, French, Chinese, Kenyan, American…}  Student\_Ancestry-Level2={Caucasian, Asian, Middle Eastern, African, Red Indian…}  Q-id\_level-3= {human} | P(Q-ID1-L2)=1/150=0.007  P(Q-ID1-L3)=1/200=0.005 |
| Q-id2 (Student\_Grade) | Suppression Student\_Grade={A+, A, A-, B+, B, B-, C+, C, C-, D+, D, D-, F}. | P(Q-ID2)=1/13=0.077 |

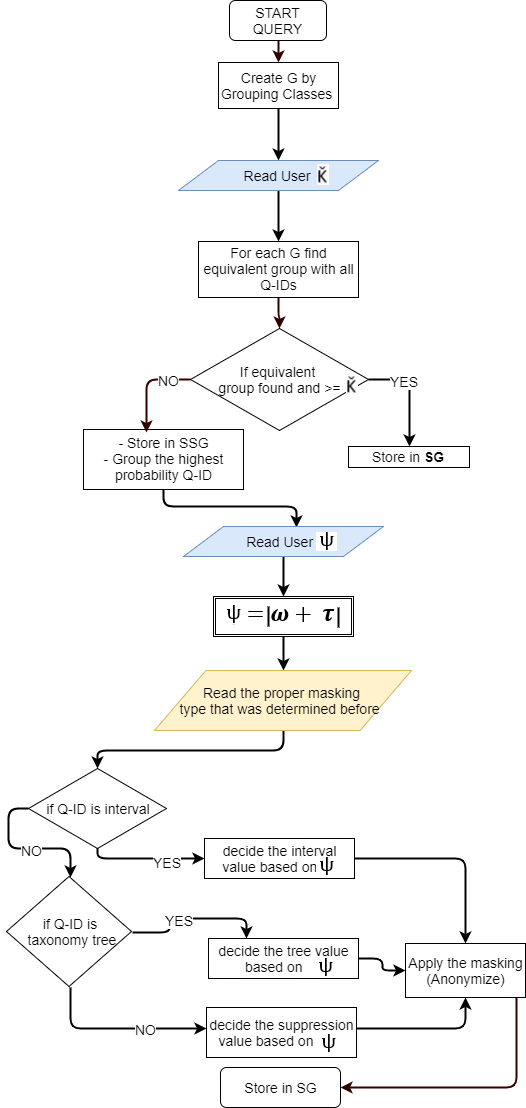
#### Anonymization Operations

The anonymization process can be divided into two main operations, the first operation is completed before the anonymization process starts, which is finding the sensitivity level ψ. The second operation is masking Q-ID values by either taxonomy tree, suppression, or discretization. The second operation relies on the first operation’s result. Any user requesting data access is subjected to an authorization level evaluation. The evaluation imposes a mathematical calculation for the sensitivity level ψ. The value of ψ is then used to calculate the degree of masking, and the number of equivalent records during the grouping process. The algorithm of anonymization process is shown in Figure 3.6. It provides a brief description about the anonymization process. The algorithm shows the parameters that the program needs to retrieve for the entire operations. The program needs to read the users and ψ, in order to compete the grouping and masking operations. Also, the program needs to know other parameters, such as the masking type, the Q-ID probability, the Q-ID names, the class attributes and others. These parameters are pre-prepared and stored in an XML file for each dataset. Hence, two major files are linked with the anonymization operations; user file and data information file.

The masking pattern is decided by the data owner, who finds the best fit mask. In some cases, suppressions can be more accurate the intervals or vice versa, and this depends on the data type. Data owner needs to be accurate on choosing the best masking pattern. The masking is applied gradually, which depends on the value of ψ. This value provides the gradualist tool on applying masking pattern. The masking pattern criterion is chosen based on the probability multiplication. The results are approximate to the maximum sensitivity level as described below:

(3.8)

Let us consider the following examples of how the masking pattern is applied. In the first example, a user was given ψ=0.04, =4, while Q-ID= {Age, Sex, Suburb}. The probability for each Q-ID is shown in Table 3.4. The created semi-equivalent records are grouped based on the highest two Q-IDs, and these are Page and Psex. Suppose that the chosen Suburb pattern is the taxonomy tree for Sydney suburbs or <Cutj>. The anonymization process relies on grouping and masking. As explained in section 3.2.3.2, the grouping of records is an iterated process. The first grouping stage filters out all the fully equivalent records, while the rest of the non-equivalent records are stored in SSG1 waiting for the next stage. In the next stage, the SSG1 group is grouped by the highest two probabilities, Sex and Age, while the Suburb Q-ID is masked by the taxonomy tree. The anonymized suburb should follow the sensitivity rule by finding a probable value ≤ ψ.



*Figure 3.6- MDSBA algorithm*

*Table 3.4- Three Q-ID attributes example*

|  |  |  |
| --- | --- | --- |
| **Q-ID** | **P(Q-ID)** | **Description** |
| Age | 0.01 | From 1 to 100 |
| Sex | 0.5 | Male, Female |
| Suburb | 0.0001 | 850 suburbs in Sydney |

*Table 3.5- Anonymization for SSG1*

|  |  |  |  |
| --- | --- | --- | --- |
| **Age** | **Sex** | **Suburb** | **Anonymized Suburb** |
| 25,25,25,25 | F,F,F,F | [Mulgoa](https://en.wikipedia.org/wiki/Mulgoa,_New_South_Wales), [Colyton](https://en.wikipedia.org/wiki/Colyton,_New_South_Wales), [Caddens](https://en.wikipedia.org/wiki/Caddens,_New_South_Wales),  Penrith | Penrith, Penrith, Penrith, Penrith |
| 30,30,30,30,30,30,30,30 | M,M,M,M,M,M,M,M | Mosman, [Balmoral](https://en.wikipedia.org/wiki/Balmoral,_New_South_Wales)  , Chowder, Obelisk, Parriwi , Taylors , Pearl , [Kirkham](https://en.wikipedia.org/wiki/Kirkham,_New_South_Wales) | Mosman, Mosman, Mosman, Mosman, Syd, Syd, Syd, Syd |
| 42,42,42,42,42 | F,F,F,F,.F | [Kensington](https://en.wikipedia.org/wiki/Kensington,_New_South_Wales), [Coogee](https://en.wikipedia.org/wiki/Coogee,_New_South_Wales), [Chifley](https://en.wikipedia.org/wiki/Chifley,_New_South_Wales), [Malabar](https://en.wikipedia.org/wiki/Malabar,_New_South_Wales), [Maroubra](https://en.wikipedia.org/wiki/Maroubra,_New_South_Wales) | ,Syd, Syd, Syd, Syd, Syd |

In Table 3.5, the first SSG1 record was anonymized by Suburb. Luckily, all suburbs in the first record were located in Penrith, hence, the generalized value became Penrith, since P(Penrith)= 1/40 = 0.025 < 0.05. In the second line, the number of grouped values was 8. Since the value of =4, then the masking can be applied on each 4 values separately. Thus, the first 4 suburbs are located in Mosman, while the rest of the 4 suburbs are mixed between different areas. Even three of them are in Mosman, but the forth suburb is from another area, so it has contaminated the masking level, and extra generalization is needed. For this reason, the masking has moved to a higher level in the taxonomy tree, which is Sydney. Mosman value is accepted because P(Mosman)= 0.33 < 0.04. In the last record, the suburbs are located in one area, Randwick, but P(Randwick)= 0.47, which is > ψ . Therefore, the masking cannot be with Randwick, so Sydney masking is chosen. After applying the masking process on the SSG1 records, the anonymized data is stored in SG group.

In the second grouping stage, the rest of the SSG1 records are grouped again with the highest Q-ID probability. This implies that records will be grouped by Sex only. The rest of the two Q-IDs, Age and Suburb, will be anonymized. The probability factor of both Q-IDs must not be above the ψ. Table 3.6 illustrates this kind of anonymization. This grouping will result two records only, for male and female.

*Table 3.6- Anonymization for SSG1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Anonymized Age** | **Sex** | **Suburb** | **Anonymized Suburb** |
| 29, 31,31,32,… | [25-35[,[25-35[, [25-35[,[25-35[ | M, M, M, M,… | [Lansvale](https://en.wikipedia.org/wiki/Lansvale,_New_South_Wales), [Cabramatta](https://en.wikipedia.org/wiki/Cabramatta,_New_South_Wales), [Carramar](https://en.wikipedia.org/wiki/Carramar,_New_South_Wales),  [Yennora](https://en.wikipedia.org/wiki/Yennora,_New_South_Wales),… | Fairfield, Fairfield, Fairfield, Fairfield,… |
| 22,25,32,32,41,… | [20-35[, [20-35[, [20-35[, [20-35[, [40-45[,… | F, F, F, F, F,… | [Kensington](https://en.wikipedia.org/wiki/Kensington,_New_South_Wales), [Coogee](https://en.wikipedia.org/wiki/Coogee,_New_South_Wales), [Chifley](https://en.wikipedia.org/wiki/Chifley,_New_South_Wales), [Malabar](https://en.wikipedia.org/wiki/Malabar,_New_South_Wales), [Maroubra](https://en.wikipedia.org/wiki/Maroubra,_New_South_Wales),.. | Randwick, Randwick, Randwick, Randwick, Randwick,… |

As noticed in Table 3.6, the anonymized age is masked by intervals, and the table shows only a small part of data. For security purposes, the minimum interval is set to 5. The probability factor for the first record is P(total)=P(Fairfield) \* P(Age)= 0.037 \* 0.1= 0.0037, which is smaller than ψ. The interval was chosen based on the Age values. The first value of the Age record was 29, and the forth value was 32. To fulfill k-anonymity requirements, a minimum of 4 records must be equivalent. Thus, the interval has included the first four values. In the second record, the Suburb masking of Randwick was not possible in the previous table. However, in this stage the factor of both Q-IDs has enabled a better masking level in the Suburb attribute. The comparison between Table 3.5 and 3.6 illustrates the difference between both Suburb records. In the first table, the Suburb anonymity was generalized to (Sydney), while in the second table the generalizing was set to (Randwick). This is related to the probability factor decrease.

It is clear that each record of the previous table 3.6 contains a long array of values. This is due to the Sex grouping. This kind of grouping separates all the dataset into two compressed records only. Hence, anonymizing each one of these arrays is not an easy task to tackle. Here are some facts we need to remember before masking this array. The masking pattern is chosen based on the sensitivity value ψ. The algorithm can be used to determine the closest masking pattern. The general algorithm is illustrated Figure 3.6.

Anonymization methods apply data distortion using masking operations. Masking implies a taxonomy tree, suppression or discretization. The taxonomy tree is the key anatomy for data masking. It implies hiding special information by generalizing them. For example, if the data contains person’s suburb as ‘Sydney’, then the taxonomy tree contains Australia (country) 🡪 NSW (state)🡪 Sydney (city). In Bottom-Up Generalization, the masking of the first cut is NSW, and the second cut is Australia. Discretization means replacing numerical values with a single interval and is denoted by *Int.* The interval deals with numerical data, where a set of numbers is presented by two numerical values for start and end. Finally, suppression of value means replacing all its relevant values with the sign \* or other characters. This operator is denoted by ‘Sup’ [21]. Also, suppression can be presented by other values, such as ‘any, or ‘person’. Some data can be only masked by suppression, such as person’s gender {Male, Female}. Other data can be only anonymized by using interval or taxonomy tree, as shown in Table 3.7. The three masking patterns are presented as <∪Cutj ,∪Supj ,∪Intj> [22]. The sensitivity value of the attribute S is calculated based on the user access level and other factors. This sensitivity value is the milestone that determines how the masking pattern will be applied on Q-IDs. Firstly, the Q-ID data type is chosen, as shown in Table 3.7. Secondly, the masking pattern is being implemented on the chosen data type. This imposes three masking tools for anonymization, which are; taxonomy tree level for ∪Cutj pattern, interval distance for ∪Intj, and the number of suppressed digits for ∪Supj pattern.

*Table 3.7- The masking Pattern for some data types*.

|  |  |
| --- | --- |
| **Masking Pattern** | **Data Type** |
| ∪Cutj, ∪Intj, ∪Supj | Date |
| ∪Intj, ∪Supj | Integer |
| ∪Cutj | Polynomial |
| ∪Supj | Binomial (Ex. Yes / No) |
| ∪Intj | Real |
| ∪Supj | Text |

The user access level influences the chosen masking pattern for Q-ID. In Table 3.7, some data types can be distorted by any of the three masking patterns while others permit one or two patterns only. For example; postcode can be masked by either one of the three patterns while gender can only be suppressed by ∪Supj. Also, masking methods imply different security levels, as explained in the next section. Two main types of masking are introduced in MDSBA, these are

#### Taxonomy Tree Masking

On grouping records, a long array is created containing a large bag of data records. If the array was a set of text values that require ∪Cut or taxonomy tree. Then the algorithm reads the masking patter, and applies the masking accoringly. In the taxonomy tree masking, the algorithm follows these steps;

|  |
| --- |
| **Input**: list={a1,a2,a3,…..an}, a∈ R: list is in ascending order  **Input**: read the sensitivity value ψ, ownership , and k; |
| 1. Read all data values and store them in array arr1 2. Read the probability of taxonomy node and find the probability factor for the other Q-IDs 3. If the factor <= ψ 4. Generalize all array objects to the first taxonomy level, else 5. Generalize all array objects to the second taxonomy level 6. Sort arr1 based on count of objects appearances 7. Read a number of objects, check their equivalency, if equivalent store them in another array arr2 8. If reached the last objects in arr1, then read all the remaining objects in arr1 and check their equivalency, if equivalent, then store them in arr2 9. Else, generalize the objects to the third taxonomy level, and check the equivalency again 10. Iterate step 9 until reaching the root of the taxonomy tree. 11. Store the generalized objects in arr2 |

*Figure 3.7- Masking taxonomy tree algorithm*

Referring to Table 3.5 example, and recalling Figure 3.7, if the array objects for Suburb is arr1={[Roselea](https://en.wikipedia.org/w/index.php?title=Roselea,_New_South_Wales&action=edit&redlink=1), Mosman, [Balmoral](https://en.wikipedia.org/wiki/Balmoral,_New_South_Wales) , Chowder, Obelisk, [Cowan](https://en.wikipedia.org/wiki/Cowan,_New_South_Wales), Parriwi , Taylors, Macquarie}. Generalizing the array objects to the first taxonomy level will result in the following array arra1={Hornsby, Mosman, Mosman, Mosman, Mosman, Hornsby, Mosman, Mosman, Ryde}. Next, the array is sorted based on count of objects appearances as arr1={Mosman, Mosman, Mosman, Mosman, Mosman, Mosman, Hornsby, Hornsby, Ryde}. The array objects are now ready for equivalency check. The check reads a number of objects, check the equivalency and stores them in arr2. Here in this example, the first group in arr2 is arr2={Mosman, Mosman, Mosman, Mosman}, while the second group of objects contains 5 objects and not 4. Recalling step in 8 in the above algorithm, if reached the last object in arr1, then read all the remaining objects. Hence, if the number of the last objects < 2 X , then all objects are read and generalized. The five objects are { Mosman, Mosman, Hornsby, Hornsby, Ryde}, since the objects are non-equivalent, then the generalization moves to the next taxonomy level. The next level is given by Sydney, so the generalization will be {Syd, Syd, Syd, Syd, Syd}. The final array will contain the following objects arr2= { Mosman, Mosman, Mosman, Mosman, Syd, Syd, Syd, Syd, Syd}.

#### Discretization Masking

The discretization or interval is another masking pattern that is implemented in MDSBA. This masking pattern can be more accurate than taxonomy tree. However, choosing the masking pattern is related to the data type. Data with real, integer, or date can be anonymized by intervals. The interval requires a range of minimum (*Vmin*) and maximum (*Vmax*) values to generalize numbers within the range. The interval range can be derived from the ψ value. The interval is calculated by ∪Int =1 / ψ, where UInt ≥ 5. So if the interval was smaller than 5, then it will be set to 5. Moreover, the minimum and maximum intervals are chosen with units of 5 starting from zero, and base-five units are added on each time extension. Assigning the interval in this way is essential to protect data privacy, especially data on both interval ends. To explain this, let us consider two values of {13,14} and interval range= 2 . The interval can be written is [13-15[, which clearly tells the adversary about the original numbers. The idea behind creating a range of units is hiding the original numbers from adversaries.

The algorithm of creating the interval is illustrated in Figure 3.8. The algorithm is very similar to the previous taxonomy tree masking. The main essential part is calculating the interval for each group of array objects. It is apparent that masking process must start with taxonomy tree masking first. If masking requires more than one Q-ID, then the first Q-ID masking must be the taxonomy tree, then the interval and finally the suppression. The process concludes three main IF statements to determine the best interval for each group of objects in the array. The numerical array is firstly sorted. The Q-IDs factor is calculated and compared with ψ. If the Q-ID factor ≤ ψ, then the algorithm proceeds with the masking. The range is calculated by, so the . Eventually, we are able to find the minimum and maximum values of the interval, by referring the small array object to the low unit of 5, and adding the found interval to the minimum value, to find out the minimum value. For instance, if the minimum value is 6 and the interval is 7, then the *Vmin* =5, and the *Vmax* =15, as a result of (5 + 7=12). It is clear that both minimum and maximum values must refer to base-five rule.

After finding the interval, the algorithm loops for all objects in the array. The first IF statement finds the all objects between the minimum and the maximum values. The second IF statement is only used when the object are the last in the array. The third IF statement verifies whether the number of objects within the interval fulfill the value. The last forth IF statement creates the intervals.

To understand the anonymization algorithm precisely, let us study the following example; if a list of numerical values is given as a list={2,3,4,6,12,12,12,18,25,26,26,30}, with k\_dash()=5. The algorithm anonymizes this list as per ψ value. If ψ=0.2, then the range=1/0.2=5. The values are arranged in an ascending order. Referring to Figure 4, the program consists of four sections. The program, first, initiates some values of; length\_of\_list=10, minimum= 2 - (2 mod 5) =0, and medium =0+5 =5. In the first section of the program, the loop starts reading the first object (2), followed by the IF statement. Since the statement is true, then the counter is incremented by 1, rep=1. The next IF statement, in section 2, will be skipped, since the objects list has not reached the end of it. In the third IF statement, the algorithm jumps by 4 objects, since Remain\_to\_k\_dash= k\_dash – rep=4. So the next list number will be 12, and the other parameters will be incremented for the next loop, rep=1+4+1=6 and object=5. Also, the medium value has been updated up to 15. Proceeding to the forth statement, it is clear that the 6th list number is also 12, so the statement result is false, and the loop continues the second loop with object=6. In the second iteration, the first statement increments rep up to rep=7, while the statements two, three, and four are skipped. Similarly, the third loop increments rep by rep=8. In the third iteration, the number 18 exceeds the range, and the program skips to the forth statement. This is the first time that the forth statement is true, and the string of all\_intervals is generated by an iterated loop of 8 times, so the results will be new\_list{ [0-15[,[0-15[,[0-15[,[0-15[,[0-15[,[0-15[,[0-15[}. In this fourth statement the loop generates the new list, and both minimum and maximum are updated by minimum= 15, and maximum=15+5=20. The next loop updates the medium up to 35, so the final new list is updated by { [0-15[,[0-15[,[0-15[,[0-15[,[0-15[,[0-15[,[0-15[,[15-35[,[15-35[,[15-35[,[15-35[,[15-35[}.

|  |
| --- |
| **Input:** list={a1,a2,a3,…..an}, a∈ R: list is in ascending order  **Input**: read the sensitivity value ψ, ownership, and k; |
| **Variables definitions**  var length\_of\_list=lists.size  var minimum=list(0)-list(0)%5  var range=1 / ψ  var medium=minimum + range |
| **Processes**  //Loop to anonymize numerical  Do While object < length\_of\_list  object=object +1 //counter for objects in the list  //1. The main IF statement  IF list(object) >= minimum AND list(object) < medium THEN  rep=rep+1 //counter for objects within the interval  END IF  //2. Include the last few objects in the list within the same range  Count\_remained\_objects= length\_of\_list – object  IF Count\_remained\_objects < k\_dash THEN  medium=list(length\_of\_list-1)+(5-list(length\_of\_list-1)%5)    rep=rep + Count\_remained\_objects  object=object + Count\_remained\_objects  END IF  //3. Jump to the object that full fill k\_dash  IF rep < k\_dash THEN  Remain\_to\_k\_dash= k\_dash – rep  medium=list(object+ Remain\_to\_k\_dash-1)+(5-list(object+ Remain\_to\_k\_dash-1)%5)  IF medium – minimum < range THEN  medium =minimum + range  END IF  rep=rep + Remain\_to\_k\_dash+1  object=object + Remain\_to\_k\_dash  END IF  //4. Get the next object  IF list(object) > list(object-1) THEN  FOR I =0 TO rep    all\_intervals=all\_intervals+"["+minimum+" - "+medium+"[,"  NEXT I  minimum = medium  medium=medium + range  rep = 0 //reset the counter  END IF  END WHILE |
| **Output:** list={[A1-B1[,[A2-B2[,[A3-B3[,…,[An-Bn[} |

*Figure 3.8- Interval masking algorithm*

## Summary

This chapter reviewed a large part of MDSBA. This included general definitions, grouping and probability methods. In this chapter, it was proven that that data equivalency increases parallel with the general growth of data. This induced a need for re-considering big data anonymity techniques. Four characters were identified for any developed big anonymity method. These characters included; equivalency increase, information gain and security, parallel algorithm, and gradual access. It was explained that sensitivity-based anonymization method carried on the big data characters. This novel method is able to provide an access control for multiple level of user’s access. The main concept of MDSBA is dividing datasets into small groups of attributes (vertically), and equivalent groups of records (horizontally). These two types of grouping helps parallelization of data operations over multiple nodes. Big data needs to operate in a large scale framework such as Spark or Hadoop. These frameworks require special algorithm that are able to cope with the framework structure. Two main masking tools were discussed; taxonomy tree and discretization. The operation sequence assigns the taxonomy tree filtration first, and then discretization or interval.

The next chapter implements a practical solutions, provided with experiments. MDSBA will be further examined and compared with the other popular anonymization methods. Hadoop framework and ecosystems such as; Pig and Hive will be apply several scripts to examine the efficiency and performance of anonymizing data by MDSBA.

[1] P. Russom, "BIG DATA ANALYTICS," *TDWI RESEARCH/IBM,* vol. 4, 2011.

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

[3] N. Qing Zhang, D. Koudas, D. Srivastava, and D. Ting Yu, "Aggregate Query Answering on Anonymized Tables," ed, 2007, pp. 116-125.

[4] S. Ninghui Li, S. Tiancheng Li, and S. Venkatasubramanian, "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity," ed, 2007, pp. 106-115.

[5] C. Dwork, "Differential Privacy," *Microsoft,* 2010.

[6] K. G. Shin, X. Ju, Z. Chen, and X. Hu, "Privacy protection for users of location-based services," *IEEE Wireless Communications,* vol. 19, pp. 30-39, 2012.

[7] Y. Kwon, "Hadoop Internals," 2014.

[8] X. Zhang, C. Liu, C. Yang, J. Chen, S. Nepal, and W. Dou, "A hybrid approach for scalable sub-tree anonymization over big data using MapReduce on cloud," vol. 80, ed, 2014, pp. 1008-1020.

[9] R. K. a. B. Becker. (1996). *Adults Data*. Available: <ftp://ftp.ics.uci.edu/pub/machine-learning-databases>

[10] MatLab. (2014). *MatLab and SimuLink*. Available: https://au.mathworks.com

[11] S. G. Shui Yu, *Big Data Concepts, Theories, and Applications*: Springer, 2016.

[12] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li, "Big data processing in cloud computing environments," in *Pervasive Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on*, 2012, pp. 17-23.

[13] V. Govindaraju, *Big Data Analytics*: Oxford : Elsevier Science, 2015., 2015.

[14] H. Chen, R. H. Chiang, and V. C. Storey, "Business intelligence and analytics: From big data to big impact," *MIS quarterly,* vol. 36, 2012.

[15] A. Holmes, *Hadoop in practice*. Shelter Island, NY: Shelter Island, NY : Manning, 2012.

[16] D. B. Kirk and W. H. Wen-Mei, *Programming massively parallel processors: a hands-on approach*: Morgan kaufmann, 2016.

[17] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message-passing interface* vol. 1: MIT press, 1999.

[18] A. Lesuisse, *Data processing with Pig Latin*: Année académique 2011.

[19] S. Chaudhuri, "What next?: a half-dozen data management research goals for big data and the cloud," in *PODS '12*, M. Lenzerini, M. Benedikt, Kr, and M. tzsch, Eds., ed: ACM, 2012, pp. 1-4.

[20] M. Al-Zobbi, S. Shahrestani, and C. Ruan, "Sensitivity-based anonymization of big data," in *Local Computer Networks Workshops (LCN Workshops), 2016 IEEE 41st Conference on*, 2016, pp. 58-64.

[21] K. W. Benjamin C.M. Fung, Ada Wai-Chee Fu , Philip S. Yu, *Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques*: CRC Press, 2011.

[22] L. T. Y. Xuyun Zhang, Chang Liu, Jinjun Chen,, "A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using MapReduce on Cloud," *IEEE,* 2014.

[23] N. C. Institute. (2013). *Accessing the 1973-2013 SEER Data*. Available: <http://seer.cancer.gov/data/access.html>

[24] A. Basu, T. Nakamura, S. Hidano, and S. Kiyomoto, "k-anonymity: Risks and the Reality," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, 2015, pp. 983-989.

[25] Y. X. Rajeev Motwani, "Efficient Algorithms for Masking and Finding Quasi-Identifiers," 2007.

[26] J. L. Harrington, *SQL Clearly Explained*, 3 ed.: Morgan Kaufmann, 2003.

[27] P. Pasupuleti, *Pig Design Patterns*. Birmingham: Birmingham : Packt Publishing, 2014.

[28] H. Project. (2016). *HDFS Users Guide*. Available: https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html