# **Chapter 5: Framework for Sensitivity-Based Anonymization**

Hadoop security concurrent many modules of protection. Hadoop File Distributed System (HDFS) contains sensitive information that needs to comply with owner’s security policy, to compromise a granular access for multi-user. Multi-security levels must be available to protect the organization assets of any security breach by securing; HDFS access, data analytics processes, Hadoop domain, data, and processes/files of the operating system. Big Data ecosystem is a prone to multiple user’s access who request data analytics. For instance, organization’s alliance of business partners, contractors and sub-contractors may require to access data for data analytics. Also, each organization may conclude more than one department such as human resources, financial, engineering or medical departments. This imposes a multi-layer security system to reduce data breach [1].

Hadoop ecosystems operate with tools reside at the top of Hadoop core such as Sqoop, Flume, Pig, and Hive. These tools are essential for big data management and operations. The tools can be protected by applying security features on Hadoop core. Security features of Hadoop consist of Authentication, Service Level of Authorization, Authentication for Web Console, and Data Confidentiality. Hadoop core structure comprises three main parts; HDFS, YARN, and JobHistory. The three parts can be secured by using Authentication, known by hadoop secure mode, so each user and service needs to be authenticated by Kerberos before using hadoop services, as discussed before in Chapter 4 - section 4.2.

In data analytics, current Hadoop security features can provide a satisfactory level of protection if it was implemented and configured properly. In fact, the risk may appear on the data level by the authorized users, who should not be able to explore sensitive data. Private and sensitive data cannot be available to every authorized user. Instead, limited users should be eligible to access such sensitive data. Authorized user means a user that is permitted to access all or part of the data. Big data is correlated to large number of users, which in turn increases the percentage of exploring prohibited attributes. We need a framework that can organize and control the amount of data accessed, by implementing anonymization methods. Hadoop security features are unable to provide such a level of security to data. Hence, Hadoop security features secure data from any unauthorized access only and without providing a robust access control.

Sensitivity-Based Anonymization framework provides a security access control that mimics Role-Base Access Control (RBAC). The framework provides a granular security access for multi-user levels. This is accomplished by providing an anonymization in a fine-grained access control. Data owners may wish to provide organizations with verities of access levels. Also, each organization may contain several departments with several data interests. For instance, human resource departments may request patient’s private data such as names and contact details, while researchers may request patient’s age, sex, and health status. MDSBA framework resides in two different locations; Federation Service (FS), and Service Providers (SP). The data is managed and controlled by Data Owners (DO). The DO is considered to be trusted, while the data analyst id not trusted.

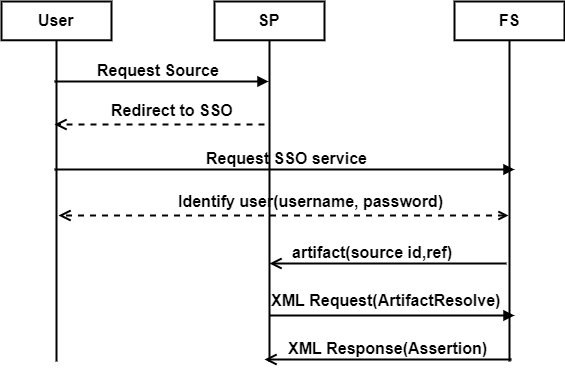
In order to implement MDSBA in a fine-grained access control, several services should be able to map between FS and SP. The mapping should match between the assigned access policy on FS side, and the access control policy on SP side. The mapping concludes three different services cooperate interactively to transfer the given access permissions from the FS to the SP. The services are: core on the FS side, initializer, and Anonymizer on the SP side. The core side is an applications that may operate at the top of the federation service/single sign-on. He core service is a database contains the details of organization’s access levels, and general security values. Before delving in these three services, it is essential to understand the transmission method between SP and FS sides.

## Security Assertion Markup Language (SAML)

SAML is an XML base single sign-on (SSO) standard that provides an authentication and authorization mechanism, with an interoperability between different security services in distributed environments. SSO invokes an ease of use access over the net for different web services. Three main objects are involved in SAML procedures these are: user, Service Provider (SP), and Federated Service (FS) [2].

SAML standard can be implemented in different scenarios, which depends on the business needs and limitations. However, all scenarios follow close similar procedures. In the mean time we focus on the most prominent scenario that initiates MDSBA framework in analytics. SAML is initiated by users, who log-in to the FS first, then request an access permission for SP. The access details and permissions are encapsulated in an XML file, known by Assertion. This file is transmitted by using SAML security tokens. Users may initially request access from the SP, if so the SP redirects the users back to the FS authentication and authorization service. SP must redirect the users request to the FS or identity provider for authentication purposes. The request is formatted in assertion notations, which is a set of XML groups that are bind using several communication protocols such as: POST, GET, Representation Sate Transfer (REST), or Simple Object Access Protocol (SOAP) [3, 4]. Developers implement GET and POST on transferring very limited parameters, therefore, current SOAP and REST are widely used communication protocols in web development. The major difference between SOAP and REST is the performance and speed. REST implements JSON file to transmit parameters as a faster option than XML. However, REST performance clearly appears in mobile applications, while in desktop applications, SOAP and REST are quite similar in performance-wise [5].

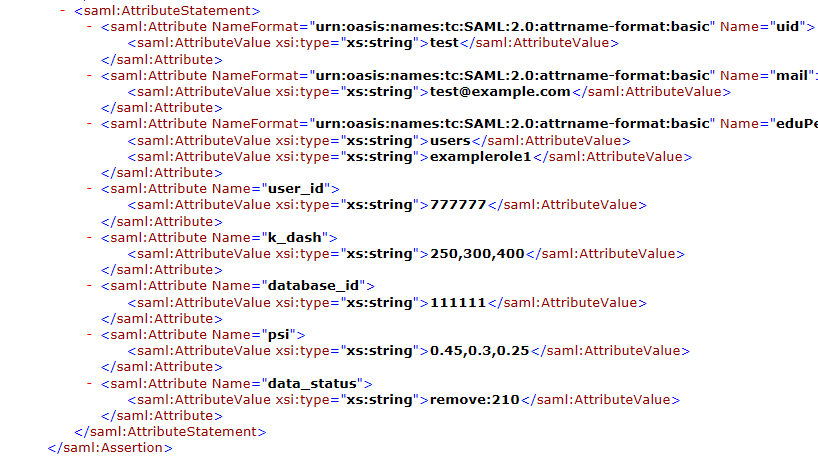
MDSBA is interoperable with two variant binding methods, and these are; front-channel bind, and back-channel bind. The front-channel bind adopts the traditional transfer methods of GET and POST. It is triggered using the web browsers between users, in FS side, and servers, in SP side. This conventional method of web communication implements the standard security of TLS/SSL, and public and private keys over TCP/443 [6]. The second binding method is usually through server-server. Its notion applies two servers communicate with each other, and without human interaction by initiating connection through a pre-defined port. The first server is located in the FS side, while the second server is located in SP side. Servers are essential to complete XML transmission through SOAP/REST servers. SOAP is used parallel with a set of artifacts. The communications procedures start by the FS artifact generator, which generates and transfers a source ID, references, and messages. This artifact is transferred by using the front-channel method or back-channel method, as described before. The SP verifies and recognizes the received artifact messages. The SP generates an XML request, known as <ArtifactResolve>. This request is transferred back to the FS. The FS generates an XML <response> and sends it again to the SP. Figure 5.1 summarizes SAML communication steps between SP and FS [7].



*Figure 5. ‑1- SAML communication steps between SP and FS*

The FS or (idP) builds a SAML XML response that contains several sections of assertions. The HTML form encrypts the XML assertions, on transmitting to the SP. The response can be sent using POST, REST, or SOAP. SAML is an XML-based single sign on (sso) standard, which provides authentication and authorization mechanism, with an interoperability between different security services in distributed environments [2]. SAML standard can be implemented in different scenarios, and this depends on the business needs and limitations. However, all scenarios follow close similar procedures as in [3, 4]. SAML 2.0 assertions divide the XML file into 8 sections as follows: response ID, Issuer ID, Status (success or Fail), Assertion ID, Signature key, Conditions, Authentication statement, and Attribute Statement. The last section contains unlimited names and values of attributes. Developers use this section to pass any authorization attributes and values [6]. In MDSBA, we need to inform the access control system with the ownership and access levels, so MDSBA algorithms calculate the sensitivity values for each permitted Q-ID group [7]. This can be embedded in the Attribute section in the XML response file. The assertion can be formatted similar to this example in Figure 5.2.

Figure 5.2 presents the XML response in SAML assertion sections. This response is transmitted from the idP or FS to the SP. MDSBA transmits some parameters through the attributes section. Other parameters are available in the SP side with XML format, and there is no need to transmit them on every user’s access. The other parameters describe the dataset details of Q-ID attributes, general attributes, class values, *k*-anonymity value for each data attribute, taxonomy trees, Q-ID probabilities, and others. The assertion XML file does not contain all needed parameters for MDSBA calculations. I only contains variant parameters that should be updated as per user’s access level. Hence, variable like, ψ, and data status are mutable and must be given to every separate user’s access.



*Figure 5.2- XML response in SAML assertion sections, and attributes section modification.*

## MDSBA and Granular Access Control

MDSBA was implemented for data analytics operating in parallel distributed operations. Hence, the gradual security anonymization should be considered in manifesting the large number of big data accesses. Since multi-user access requires fine-grained access on the data level, MDSBA provides a granular access by implementing gradual levels of anonymization. As described in chapter 3, data attributes are divided into groups of two to four Q-ID attributes. Anonymization is applied separately to each group. Data owner can decide the level of data suppression or masking on each group. For instance, Doctors do not need to know operation’s costs, and financial status for patients, hence, data owner prefers to highly anonymize financial status data and lightly anonymize health status data or group.

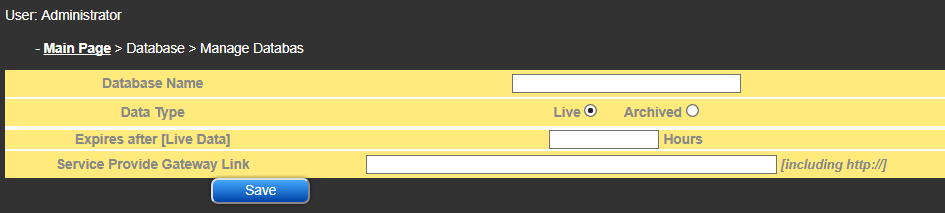
MDSBA aims to fragment data nominally to reduce the data size on accessing the anonymization software program. This can be implemented by bundling a number of Q-IDs and Classes into small groups horizontally. Medical information may contain more than one class attribute, and a considerable number of Q-IDs. Eventually, the increased number of Q-IDs may produce a massive computation cost and data overflow, which may unexpectedly terminate the anonymization program. Hence, dividing the Q-IDs into small bundles horizontally can support the performance and scalability. However, the Q-IDs are determined by data owners and can be divided randomly or logically. Dividing Q-IDs logically supports the granular access control and reduces the anonymization loss. Q-ID grouping can be divided based on user’s roles, where user’s privileges and interests varies from one role to another. User’s roles refer to organization’s departments, and each department conducts certain analytics to abstract the information of interest. For instance, HR department may needs to know the personal user’s information, while financial department focuses on the user’s financial status. Giving the right amount and value of information should be controlled to alleviate information overload and improve the security level.

### Live Data and Archived Data

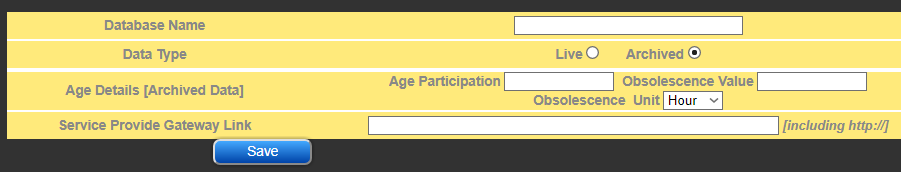
Data owner needs to assign Q-IDs and Q-ID groups, and give a proper *k*-anonymity value for each group. Choosing the optimal *k*-anonymity value will be discussed later in this chapter. Data owner, also needs to map Q-ID groups to user’s roles as explained before. In data lifecycle management, non-used data needs to be archived in order improve the storage capacity, and to provide the non-active data with a long-term retention. Eventually, data can be categorised in to two types of live and archived data. Data owner needs to determine the type of each data (live, archive). That is if data is live, then the previously mentioned aging factor in Equation 3.7 can be ignored, and the Equation 3.6 will be ψ=ω. The reason for omitting the aging factor term is data continuous update, hence, data importance will not be degraded with the time factor. Data analytics consumes an intensive computation time, which makes the real-time analytics hard to obtain. In this case, live data can anonymized with an acceptable time that is close similar to real-time. This is essential to obtain a better accuracy in analytics results. Productive database actively varies every time, and the changes of data growth varies and depends on the data nature. The anonymized copy of live or productive database may become old in few days, or even hours. This, actually, depends on the changing and growth speed of data.

In MDSBA framework, a practical solution was added for live data. This solution anonymizes data for every specific user only once. The user uses the same anonymized data on every access, and there is no need to anonymize the same data each time. This solution saves the user’s time, and improves all over performance. However, this solution cannot be implemented in live data, which is updated regularly. Hence, live data was controlled by a (life-time) parameter. If the parameter was determined by data owners, then the anonymized copy of data remains during a life-time period. When life-time period expiries, then the anonymized copy is prone for removal. When the user accesses the same dataset, the anonymized copy is verified by life-time expiry, if so, then the database copy is purged, and a new anonymized copy will be generated. For example, if the life time was determined by the data owner as 24 hours, then the anonymized copy of data will be erased if the user attempts to reuse data after 24 hours, and a new anonymization process will take a place to create a new anonymized copy.

Figure 5.3 and 5.4 illustrate the steps that data owners need to follow on creating and configuring a new dataset. The figure shows the distinction options between archived and life data that data owners need to provide. The live data requires the expiry number of hours before it is being purged, as shown in Figure 5.3. The archived data requires more parameters as shown in Figure 5.4.



*Figure 5.3-Live data options for data owner interface*



*Figure 5.4-Archived data options for data owner interface*



*Figure 5.5- Steps to configure initial values for each dataset.*

Figure 5.5 summarises the steps of preparing any new dataset for analytics. Data owners need to define the Q-ID attributes, assign them in groups, assign a *k*-anonymity value, and map them to business roles. Each Q-ID attributes must be defined by a probability value, and a masking type. Dataset is categorized as Archived or Live data, where the archived data is given the time factor parameters of; age participation, obsolescence value, and obsolescence unit. Apart from dataset preparation, data owners need to create business roles, users, organizations, and assign a set of roles to each organization. Users are crated and managed by the delegated organizations.

### MDSBA *k̄* percentage and Business roles

One of the granularity structure, in MDSBA, is assigning business roles to Q-ID groups. This is essential to formalize the access control as per business roles. Data owners prefer to determine permissions using roles, rather than dealing with Q-ID groups. Moreover, business roles can be given variable parameters depending on user’s permissions, while Q-ID groups keep their anonymity parameters regardless the user’s permissions. This imposes two separate k-anonymity parameters; k value and k̄ value. The value of k is assigned to Q-ID groups, as unchangeable fixed value to represent the optimal k-anonymity value from the grouping prospective. On the contrast, the value of k̄ is assigned to business roles, to represent the user’s access permissions. Therefore, *k̄* value is assigned to business roles that are delegated to organizations. Assigning value imposes knowing organization’s needs of business roles. One more parameter was derived from k and k̄ values relations, known as k̄ percentage (. This value is defined as the permitted level that allows a user to view data. Hence, the percentage of =100% implies that the user is given. The value of =60% implies that the user is given 60% of the k value, so if the k=100, then. Organizations are given the business role along with the value. The *k̄* is calculated as *k̄* =*.*

The anonymization process needs two parameters to apply the proper amount of masking, and these are *k* and *k̄* values. These two values are essential for calculating the value of ψ. The data owners setup each organization details, by determining the for each role. The is decided based on the service level of agreement between the data owners and the organizations. Let us study this example, which shows the sequential steps of preparing the Q-ID groups, mapping them to business roles, and assigning them to users. Table 5.1 shows a set of attributes for (Properties) data consists of {suburb, street\_name, property\_type, bedrooms\_num, bathrooms\_num, sale\_price, cost\_price, max\_offer\_price}. Such attributes can be divided into two Q-ID groups, so the first Q-ID group can be G(QID)1= {suburb, property\_type, bedrooms\_num, bathrooms\_num} with one class of {street\_name}, while the second Q-ID group can be G(QID)2= {sale\_price, cost\_price} with one class of {max\_offer\_price}. In this example, the data owner may create a set of roles for G(QID)1 = {Strata Manager, Sales Representative}, and for G(QID)2= {Construction Contractor, Strata Manager}. The given roles determine the data of interest for each role. However, users can be given more than one role, and each role can be mapped to many Q-ID groups, as shown in Figure 5.6.

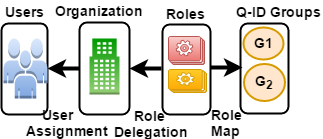
Table 5.1. Illustration example of Properties data for roles and Q–ID groups

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G(QID)1** | **G(QID)2** | ***k*1** | ***k*2** | ***k̄* \_perc for Roles [ABC company]** |
| Suburb | cost\_price | 500 | 700 | Strata Manager (70%)  [Mapped to G(QID)1 & G2(QID)2] |
| property\_type | sale\_price |  |  | Sales Representative (80%)  [Mapped to G(QID)1] |
| bedrooms\_num |  |  |  | Construction Contractor (10%)  [Mapped to G(QID)2] |
| bathrooms\_num |  |  |  |  |
| **CLASS:**  street\_name | **CLASS:**  max\_offer\_price |  |  |  |

Let us refer back to the previous property example, as shown in Table 5.1, by considering a user U1 belongs to ABC Company. The company was given three roles of {Strata Manager, Sales Representative, and Construction Contractor}. The company is given three roles to delegate users any role of the three given roles. The user U1 was given two out of three roles. The owner assigned *k*1=500 for the G(QID)1 group, and *k*2=700 for the G(QID)2. Moreover, [Strata Manager]=0.7, and [Sales Representative]=0.8 for ABC company. The user U1 permission will be given based on the percentage of . Hence, the first *k̄* value for the first group is *k̄* = 0.8 × 500 = 400. In the second Q-ID Group, Strata Manager is the only give role that is mapped to G1, so *k̄* = 0.7 ×700 = 490. The final result shows the user U1 with the following *k* and *k̄* ordered values U1(*k*, *k̄* )={(500,700),(400,490)}.

In the previous example, since there are two Q-ID groups available in this dataset, then two-pair of integers will be created in order to be transmitted to the SP by SAML. The number of created integers must be equal to the number of Q-ID groups. If the number of Q-ID groups are three, then three-pair will be transmitted and so on. The pair consists of values for each Q-ID group. The pair represent the permission decision for each group, that is, if, then the Q-ID group is suppressed, else it is permitted. The SP-Gateway interprets this notation accordingly. For example, let a user U2 is given the following; U2()={(80,80),(100,80)}. This notation is interpreted on the SP side as; G1[suppression], and G2[permission]. In another example, if a user U3 was assigned to the three available roles, then the two-pair for G(QID) are given as: *k̄* for Strata Manager is *k̄* = 490, for Sales Rep. is *k̄* = 400, and for Construction Contractor is *k̄* = 630. G(QID)1 is given the lowest *k̄* value, which is *k̄*= 400. Similarly, the G(QID)2 is given the lowest *k̄* value, which is *k̄*= 490. The final result is transmitted to the SP as the follow: U3()={(500,400),(700,490)}, because Strata Manager role is mapped to both of the Q-ID groups.

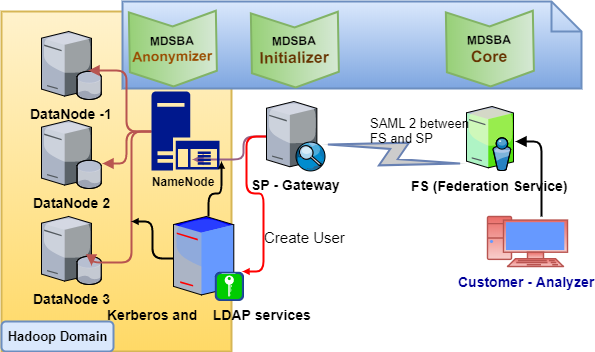
The organizations are delegated to control their own user’s permissions and roles assignment. However, organizations cannot exceed the pre-determined given values by the data owner. Eventually, users inherit these percentages from their own organizations. Also, each organization is given limited roles with for each role. Figure 5.6 illustrates the G(QID) to role mapping with the organizations intersections. For instance, data owner decides to give organization O a permission to analyse G1 and G2 only, then roles R1 and R2 can be given accordingly. Moreover, G1 and G2 can be assigned to one or more role(s). Delegating organizations, to assign roles to users, is a power technique that may reduce the management overhead by data owners.



*Figure 5.6- Organization-Roles-Groups mapping*

### MDSBA Three Services

MDSBA framework starts from the Federation Service processes and ends with the user’s analytics processes. The framework consists of three main services; core, initializer, and anonymizer. The core resides in the Federation Service side, while both of initializer and anonymizer reside in the Service Provider side as shown in Figure 5.7. MDBSA framework consists of four groups servers, these are: FS, SP-Gateway, Kerberos and LDAP, and Hadoop domain. Customers are defined as any user who attempts to access Hadoop domain for data analytics, and can be the data owner or a customer from any external organization. However, the external customer must be approved by the data owner. Both of FS and SP-Gateway operate SAML v2 servers for data transmission between SP and FS. Also, SP-Gateway contains the initializer service, which remotely creates the anonymization script. The anonymization script is executed in NameNode server. In Hadoop domain, a domain server contains LDAP and Kerberos services to provide security and user’s authentication and authorization within Hadoop domain. Users are authenticated twice with two separate authentication accounts. The initial authentication is located in the core service, while the second authentication is especially for Hadoop domain and located in anonymizer service.



*Figure 5.7- MDSBA three main services, core, initializer and anonymizer*



*Figure 5.8. Sequence Diagram for MDSBA framework*.

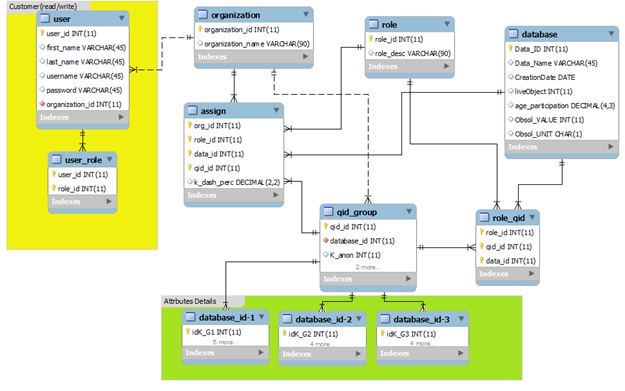
Figure 5.8 illustrates the sequence diagram for MDSBA three services; core, initializer and anonymizer. The user requests an access to Hadoop domain for analytics tasks. The FS authenticates the user and determines the belonging organization and roles attached to it. The service, also, determines the user’s own roles assigned by his/her own organization. The generated XML request by SAML contains attributes with information about user id, data id, data status and the other sensitivity parameters. SAML, thereafter, takes over the communication procedures between FS and SP. The XML assertion, then is transferred to the SP-Gateway, which in turn receives the request, and retrieves the data information with the help of the data id that was received from the assertion XML. The SP-Gateway contains a set of files as described in Table 2. One of the essential XML files is definder.xml, which contains the necessary information about the requested data. The Core service is competed when SAML takes over.

As shown in Figure 5.8, the algorithm illustrates both services of Initializer and Anonymizer. The Initializer service starts by creating the Pig script, which is generated by collecting information from parsing two XML files, definder.xml, and SAML assertion file. The Initializer service, also, verifies the user’s availability from the LDAP / Kerberos server. The FS provides a user id, which is considered as the username on the SP side. Hence, the username that is generated from the FS user id, will be created in the LDAP server, if it wasn’t created before. In the third service, or the Anonymizer service, data is copied to Hadoop domain, and specifically to the NameNode. If the username was created before, then the Anonymizer service verifies weather to purge the previously anonymized data or keep. If data was purged, then the Pig script is triggered and a new anonymization process starts. Later the user is provided with all access details including username, password, and data location of input and output paths.

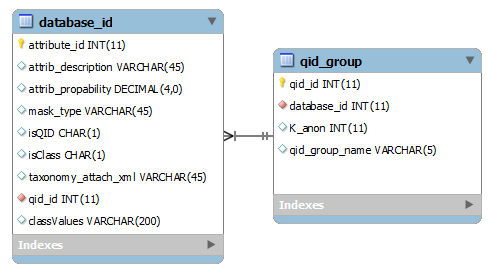
#### Core service

The core provides the initial authentication and authorization for multi-access user’s level. The core service consists of a data schema and user interfaces to update MDSBA users and datasets details. User’s details include: user name, login details, organization’s names, general roles, and roles assigned to the organization. Data owner configures initial settings for each targeted dataset that users wish to access and analyse. The core service initial setup are shown in Figure 5.5. The aim of the setup is identifying the Q-IDs and classes in the data schema. Each two to four Q-IDs and one class are assigned to one group. The chosen Q-IDs group and a class are not necessarily found in one table. The created roles conventional names should mimic the user’s actual roles.

Figure 5.9 illustrates the Entity Relationship Diagram (ERD) for MDSBA core schema. The schema is located in FS side. The relationship diagram describes a basic information data diagram to store organization’s users, and roles. The diagram presents a (database) entity, which contains the {data\_name, creationDate, liveObject, ageParticipation, obsol\_VALUE, obsol\_UNIT}. The interface of this entity is shown in Figures 5.3 and 5.4. The attribute of (liveObject) presents the expiry time in hours for the live data. Hence, if data type is archived, then liveObject value is zero. If data type is live, then ageParticipation, obsol\_VALUE, and obsol\_UNIT are set to zero. As shown in Figure 5.7 ERD, the data owner is responsible about initializing the objects of: database, assign, organization, role, and qid\_group. The entity (role\_qid) is an associative entity to resolve the many-to-many relationship. The (assign) entity is also another associative, which contains an essential attribute of k̄ percentage. Other entities are delegated to organizations to manage their own user’s details including; name, authentication, and roles assigned to users. Actually, business roles are given to the organizations, and not directly the users. Therefore, users do not have the privileges to obtain any business role beyond his/her own organization’s roles.

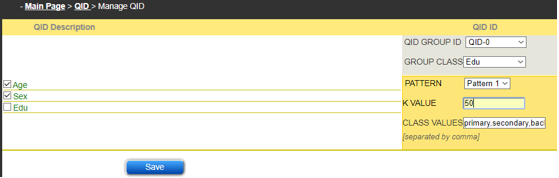


*Figure 5.9-The core part of MDSBA framework.*



*Figure 5.10-The core part of MDSBA framework*

Figure 5.10 presents the data owner grouping for Q-ID attributes. The discrete rectangular line shows a one–to-many relationship between Q-ID groups and a detailed description for each attribute in the dataset. Each created dataset imposes an auto creation for a new table named with the database\_id. The new table lists all the dataset attributes, with full details of: description, probability, and masking type (taxonomy, integer, or suppression). The attributes (isClass) and (isQID) are Boolean values. Any data attribute can be either a class, Q-ID, or an ordinary attribute. Hence, three possible value-pair are assigned to both attributes; {(0,0), (0,1), or (1,0)}. Figure 5.9 describes this relationship between qid\_group, and the database\_id entity. One of the database\_id attributes is classValues. This attribute lists all available values, if the attribute was chosen as a class. These values are essential during the late anonymization stages. Figure 5.11 shows one of the administrator’s interfaces that is able to create Q-ID groups. The main parameters are: choosing the Q-ID attributes, the k value, the class attribute, and the classValues.



*Figure 5.11-Interface to create G groups of Q-IDs, classes, and classValues*

##### Keep or Remove Decision

Core service generates several XML files for transmitting them to the service provider side. The services include assertion.xml, definder.xml, and taxonomy trees files. The assertion file is generated by SAML server. The generation process requires some parameters read from MDSBA data schema. The service provider needs to know whether to keep the anonymized copy of the dataset or to remove it. The SP offers a service to purge the anonymized copies every certain period of time. This service needs to know the exact expiry time for the anonymized copies. To do so, the core service examines two statuses of data types, when users attempt to access the service provider datasets. Live (production) or archived data are distinguished before determining the expiry time of the anonymized copies. Both cases are related to figure 5.3 and 5.4. The live data is related to the expiry period setup by the data owner, while the archived data is related to the obsolescence value setup by the data owner as well. The core service generates the assertion.xml file as per these available information.

The assertion xml file contains one attribute, known by data\_status. This attribute presents two values for the data status (keep or remove: time period). If it was the first time for a certain user to access the specified dataset, then then initial value is presented by (remove: E(D,U)), where E(D,U) denotes the time period before the anonymized dataset expires. This value is interpreted in the SP side as: a new anonymization process takes a place and the anonymized copy will last for the time period provided. For instance, if the value provided was [remove:6], then a new anonymization process will start, and the expiry time for this copy is 6 hours. If the same user has accessed the same dataset after 4 hours, for instance, then the status will be replaced by [keep]. Notice that the status [keep] does not need to be presented with the time period E(D,U). If the user has accessed again on the next day, then the status will be replaced by [remove: 6]. The SP side will look at the status and store the time period added to the status [remove: time period].

##### Initial Access

The FS determines keep or remove, by investigating the latest user’s access for the requested dataset. Initially, if the user never accessed that dataset and this was the first attempt to access this dataset, then an initial value is created. Let us formalize the expiry time for each dataset. Suppose that the current date/time is denoted by, and the expiry period given by the data owner is denoted by. The expiry time period for the specified dataset D, and the user U is denoted by, as mentioned earlier. For the user’s initial access when data type=live, the equation used is:

(1)

For the user’s initial access when data type=archived, the obsolescence value Ø is replaced by the expiry period. Moreover, the equation depends on the date and time the dataset D was initially uploaded, which is denoted by. Thus, the equation used is:

(2)

The FS calculates the value of, and transmits it parallel with the data status of [remove: ]. On the other side, the SP interprets this value differently. The SP stores value in the log file to purge the created dataset after the expiry period. For instance, if the data owner assigned (=74), and the user’s access date and time was 5-Dec-2017 at 13:10. Initially, the FS will transmit the value of [remove: 74]. The SP will purge the anonymized data at the end of the expiry time, which is 5-Dec-17 13:10 + 74 = 8-Dec-17 at 15:10. The 74 hours conclude three days and two hours. Therefore, the dataset will not be available after 8-Dec-2017, 15:10. During the past three days, any access attempt will provide a data status of [keep].

##### Subordinate Access

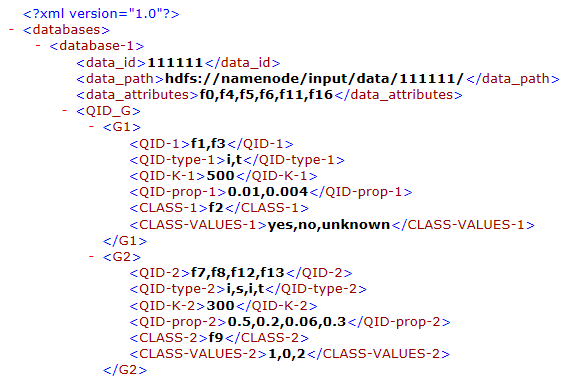
The FS determines keep or remove, by investigating the latest user’s access for the requested dataset. If it was found that the user has already accessed the specified dataset, then the value is calculated for data type=live as:

(3)

If data type=archived, then the obsolescence value replaces the time period. The FS investigates the latest user’s access through the log file, if it was found that the user has already accessed the specified dataset, then the expiry date is calculated by Equation 2.

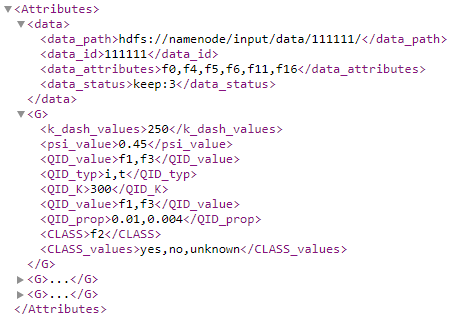
#### Initializer Service

This is the second component of MDSBA framework. The service operates in the SP side, and outside Hadoop domain. SP should provide a gateway service, which is considered to be the first entry level for users who wish to participate in data analytics. For security purposes, the gateway server should be isolated from Hadoop domain. This service provides initial documents to prepare data for anonymization. SP needs to provide a valid path in the SP-Gateway server for data owners. Each uploaded data on SP servers must be provided with essential set of documents to support anonymization process. Data owners need to upload these files over the given path in the SP-Gateway. The files consist of two XML files, one ecosystem scripts such as pig or hive, and the anonymization software program such as Java Jar. One of the XML files is the definer.xml, which defines Q-ID Groups and their related Q-IDs and the other attributes. The format of this file is described in Figure 5.11. The file contains database Id, path, attributes, and Q-ID groups, types and classes. These attributes are needed to provide complete information about data on the SP side. The Q-ID types are given the letters (i,s,t) to denote the interval, suppression, and taxonomy tree consequently.



*Figure 5.12. Definder.xml file demonstration.*

The file (definder.xml) was generated from the core service. All the information are stored in the data schema and can be easily generated as per dataset. The file consists of Q-ID groups created from data schema as shown in Figure 5.12. SP-Gateway server communicates with the FS throughout SAML v2 server. On user’s login, the core service generates another XML file of SAML assertion (assertion.xml), which contains login and anonymization information in the attributes section of XML file, as shown in Figure 5.2. The XML assertion file is unlike the definer.xml, since it is generated on each time a user requesting access, while the definder.xml is transferred to the SP-Gateway only on dataset alteration. The attribute section of definder.xml file contains the following information: user id, database id, k̄ values, sensitivity level ψ, and data status. Data status consists of two different values; keep or remove. The data status defines the previous anonymized copy and its validity for the same user. This decision is made on the FS side, and based on the last login time by the user. The decision is made by comparing between the last login, and the obsolescence value for the archived data or the liveObject for the live data.Table 5.2 illustrates the essential available files for each dataset on the SP side, as a part of the initializer service in MDSBA framework. Table 5.2 shows the output files of pig script, and data\_id.xml. the data\_id file is very important for Java programs during the anonymizing process. Java files, such as ADJUST.java, SSG\_P1.java, SSG\_P2.java need some parameters to execute the program. The file data\_id.xml is a combination of definder.xml and assertion.xml files. The file is generated by the initializer service, which contains the following attributes: All information contained in definder.xml + some information abstracted from the SAML assertion including (data\_id, user\_id, k̄, ψ, data\_status {keep, remove}. Figure 5.13 presents an example of data\_id.xml file. The example shows complete details about the dataset id, path, attributes, status,, and details of each group. The data status is presented by keep: 3, which indicates the dataset can be kept for the next three hours. Similarly, if the status showed remove: 8, then the dataset will be purged immediately, and the new anonymized copy will continue valid for the next 8 hours. The unit used in the data status is hours. If data type is not live (archived), then data will not be purged while the copy age is smaller than the obsolescence value. However, usually the obsolescence value is much larger than live time period. Therefore, the data status in data\_id.xml file is possibly large, such as keep: 4320, which indicates that the obsolescence value is 6 months, and data will be purged after six months.



*Figure 5.13- Sample of data\_id.xml file*

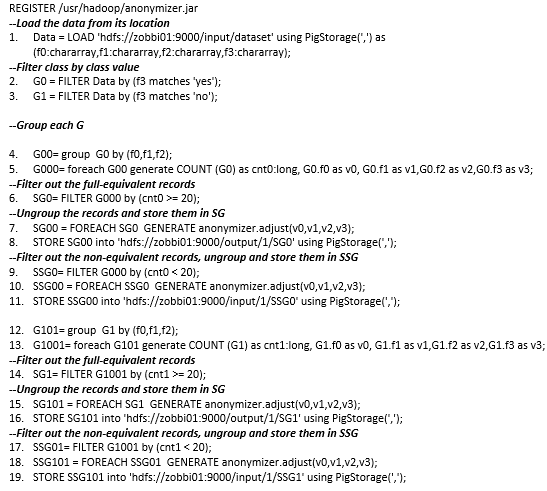
Table 5.2. List of files used by the initializer service

|  |  |
| --- | --- |
| **File name** | **File description** |
| **MAIN FILES** | |
| definder.xml | Generated by core service, and uploaded by data owner once, or on data alteration. It contains data information including:   * Attributes named by f sequence: * Q-IDs named by f sequence * Q-IDs anonymization type { taxonomy: t, interval: I, suppression: s}, * K value for each Q-ID group * Q-IDs probabilities * Class named by f sequence * Class values |
| assertion.xml | Generated by the core service, which contains the following attributes:   * All information contained in definder.xml + some information abstracted from the SAML assertion including (data\_id, user\_id, *k̄* , ψ, data\_status {keep, remove} |
| **OTHER FILES** | |
| User\_id-i.pig | Set of pig scripts files. Generated by the initializer service |
| Anonymizer.jar | Contains six java files: ADJUST.java, SSG\_P1.java, SSG\_P2.java, NG\_P.java, SUP\_P.java, OBV\_G |
| Taxonomy trees [fi.xml] | Generated by core service, and uploaded by data owner once, or on data alteration. Each file represents one Q-ID attribute of taxonomy tree type. This files is used by Java files during the anonymization process. |
| Data\_id.xml | Generated by the Initializer service, which contains the following attributes:  All information contained in definder.xml + some information abstracted from the SAML assertion including (data\_id, user\_id, k̄, ψ, data\_status {keep, remove} |

##### Generating Pig scripts in Initializer Service

The Pig script is essential to anonymize data before permitting users access. The script is generated based on the definder.xml, and SAML assertion.xml files. These two files provide the essential parameters to create a set of Pig scripts files. The anonymization script will be transmitted to Hadoop domain. The initializer service creates one Pig script file for each Q-ID group. As described before, Q-ID group may contain a number of two to four Q-ID attributes. Hence, the number of generated scripts depends on the number of Q-ID attributes in the group.

The initializer service reads definder.xml file to obtain data information including data location, Q-ID attributes, k values, status, probabilities, and classes. The initializer first determines the data status, whether it is in (keep) or (remove) status. If data is in keep status, then no any anonymization occur and the user can login to his/her previous HDFS directories. Else, the old anonymized data will be purged, and a new anonymization process will start. Second, the initializer creates the scripts line by line, starting by registering the Jar file, assigning the load location, filtering, creating the user HDFS directories, grouping and anonymization by UDF Java. Figure 5.12 illustrates an example for Pig script created by the initializer service.



*Figure 5.14- Created Pig script file by the initializer service*

Figure 5.14 describes part of the generated Pig script for three Q-IDs. The full script is attached in Appendix A. The script filters the class value as per two values of (yes, no). Hence, two G groups are created, and each group is aggregated by three Q-ID’s, then by two Q-IDs, and finally by one Q-ID. In Pig Latin, it is possible to name attributes with (f) letter instead of using the attribute name. Before generating the Pig script, the initializer initiates two HDFS directory paths for the user. However, the directories are not created so far, but they should be named with the standard of input and output as follows:

*hdfs://namenode:9000/[data\_id][user\_id]/input,*

*And hdfs://namenode:9000/[data\_id][user\_id]/output*

Two HDFS directories are created for each user, and two Pig scripts files will be generated based on the given information. The HDFS path consists of the NameNode, followed by the port number, the data ID, and the user ID. The default port number is usually 9000 or 8020. This given path will be created in the anonymization service stage, as it will be described in the next section.

#### Anonymizer Service

This service operates between the SP-gateway and the NameNode, and specifically in both of the gateway server and the NameNode server. The service process is located in SP gateway to copy and execute scripts in NameNode server remotely. The process of the service copies the output files from the initializer service. The files include XML, Pig, and Jar files, as shown in Table 5.2. The anonymizer service executes anonymization scripts in Hadoop domain. The domain is controlled and managed by LDAP and Kerberos server, which provides authentication and authorization services of domain user’s access. Also, Kerberos is enabled to support tickets granting for Hadoop services, which creates Hadoop secure mode.

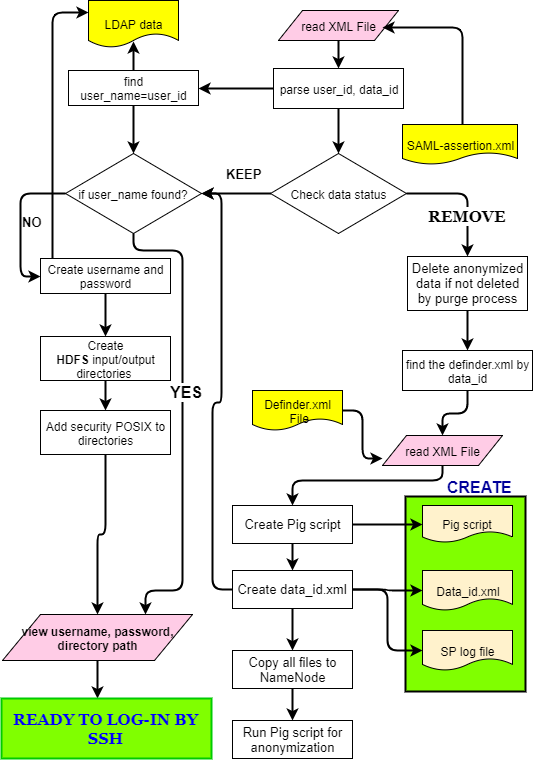
The anonymizer service executes shell commands by implementing web programming. This technique is secure and possible with the new web programming extensions and libraries. MDSBA adopted PHP programming language, with the support of phpseclib, php-devel, php-pear, and libssh2-devel extension packages. The packages perform properly with the SSH2 [8]. SSH2 can access other devices to execute commands via an interactive shell with read and write permissions, as shown in Figure 5.13. The command script is contained in a secure web page to protect the secure domain sudoers account related to the executed shell commands. The web page can be either by a trusted certificate or even by a self-signed certificate. The used account should have full privileges over HDFS storage directory and part of LDAP sudoers [9]. There are subtle differences between LDAP sudoers and local sudoers, so the used account in the shown script must contain an LDAP domain account, as shown in bold [10]. The script executes the Pig script remotely, as shown in Figure 5.13 script. Before executing the script, the previously mentioned files, of scripts and XML files, must be copied to NameNode server. The command in SSH2 is given by *$command="scp /var/www/html/".$user\_name.$databas\_id."/\*.\* /$NameNode\_server";*

|  |
| --- |
| <?php  include('Net/SSH2.php');  $ssh=new Net\_SSH2('namenode.fullname');  **$ssh->login('hadoop','password') or die ("Login failed");**  ------------------------------------------------------------  $ssh->exec("hadoop fs -mkdir /[data\_id][user\_id]/input");  $ssh->exec("hadoop fs -mkdir /[data\_id][user\_id]/output");  $ssh->exec("hadoop chown -R user\_id hdfs://namenode:9000/[data\_id][user\_id]/input/");  $ssh->exec("hadoop chown -R user\_id hdfs://namenode:9000/[data\_id][user\_id]/output/");  $ssh->exec("pig –x mapreduce script-1.pig");  ?> |

*Figure 5.15- PHP sample for anonymizer service*

MDSBA framework maps the user\_id of the Federation Service to Hadoop domain through LDAP service. The same user\_id is created in LDAP domain, if it was not created before. The anonymizer service verifies the user\_id availability with the LDAP service. If the user\_id is not available, then create a username and a random complex password. For security reasons, the password can’t be mapped with the FS password. Hence, a new created password is required to authenticate the user on accessing Hadoop domain. So the user can access the NameNode though SSH. If the user\_id has already been created, then verify the anonymized data weather to delete it or keep it, so the user can use the same previous authentication details. For the first time access, a shell command line is triggered to create the input and output HDFS directories, and finally run the Pig script, as illustrated by Figure 5.15 in PHP programming.

Figure 5.16 illustrates the SP anonymization algorithm. The procedures describe both initializer and anonymizer services to permit users accessing anonymized copies of the dataset. The procedures are summarised by parsing data ID and user ID from assertion.xml file. The assertion file provides some needed information to build-up the Pig Latin script, which creates anonymized copies of datasets. The first conditional statement, in the algorithm, verifies the whether the username was created before or not. The process interacts with the LDAP server, and fetches the username by the user\_id, where username=user\_id. If the username was not found, then it will be created by using the user\_id as a username, and a randomly created password. Next, the HDFS directories should be created and given the permission. Creating the user’s directories requires the knowledge of user\_id and data\_id. The initializer service needs to read two XML files, definder and assertion, to prepare the needed documents for the anonymizer service. The initializer first creates three main files of; Pig script, data\_id.xml, and SP log file. The data\_id.xml and the Pig script are passed to the anonymizer service, which in turn, copies the essential files to a temporary file in the NameNode of Hadoop domain. Finally, the Pig script is executed so the anonymization copy of dataset is ready at the completion of the anonymization process.



*Figure 5.16- Initializer and Anonymizer algorithms*

|  |
| --- |
| Algorithm for Anonymizer service |
| 1. Login automatically by Hadoop username and password 2. Read the user\_id and check if it is available in the LDAP domain 3. If not found then create a new username=user\_id and a random password for the user 4. Create HDFS path for input and output by using Hadoop authentication 5. Give permissions to the username=user\_id for HDFS input/output folders 6. Create temporary folder in NameNode server 7. Copy all needed files to the temporary folder 8. Run the Pig Latin script 9. The user log-in to the anonymized copy by using SSH interface |
| Output: Anonymized copy of the original dataset |

*Figure 5.17- Algorithm for Anonymizer service.*

Algorithm 5.17 shows two authentication levels by user, to access Hadoop domain. This is essential in securing Hadoop domain. The first authentication process is required by the FS to find out the user’s authorization level. The second authentication process is needed for SSH access. The password authentication is essential on accessing NameNode server through SSH interface. Users are able to use SSH to access Hadoop service at any time to submit analytics queries, and without going through the long process of the first sign-in. However, this shortcut may not provide users with the latest live dataset. The solution for this issue is creating a continuous process to purge the anonymized copy of the dataset. The purge process continuously reads the SP log file, in order to determine the purged data.

##### Service Provider Log File and Purge Process

The log file is a major component for any server service. It is a primary tool on tracking all activities occur on servers. For this reason, MDSBA creates its own service provider log files. One of the primary log files is created for a continuous process, known by purge process. The process mission is purging all expired anonymized datasets to utilize storage capacity. Original datasets may be live data that is updated regularly. They may be archived datasets with a time factor. The time is an essential parameter to determine the anonymization and access level. MDSBA reduces the archived data masking, to allow more gained information, parallel with the time factor. Users should not use the same anonymized copies of datasets at all times. The current anonymized copies may not be valid after a certain period of time. Therefore, purging datasets and creating fresh anonymized copies is needed. The created log file must record the user ID, the data ID, the anonymized data path, and the expiry date/time, as shown in Table 5.3.

The FS side interprets the attribute data\_status of assertion.xml file. The attribute notation is presented by [keep/remove:]. The anonymizer service will not take any action when the status=keep, while few steps are considered if the status=remove. The remove status is always attached with expiry period of time, therefore, the anonymizer service registers part of the details in the log file, as shown in Table 5.3, and proceeds to the next following steps. The anonymizer, firstly, checks if the previous dataset was purged, or not. If it was not purged, then the anonymizer process will purge the previous anonymized copy. Secondly, a new anonymization process will be initiated.

Table 5.3- Log file for the purge process

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **User ID** | **Data ID** | **Anonymization Path** | **Expiry date/time** | **Last purge attempt** | **No. of Attempts** | **Purge Result** | **Purged (yes/no)** |
| 123456 | 111111 | */111111123456/input* | 05-01-18  15:20 | 05-01-18  18:10 | 4 | Fail | No |
| 321215 | 111111 | */111111*321215*/input* | 12-03-18  11:05 | 12-03-18  13:30 | 2 | success | Yes |
| 123456 | 222222 | */*222222123456*/input* | 25-03-18  01:20 | 24-03-18  03:30 | 3 |  | No |

Table 5.3 receives entries from two separate services. The first four attributes are added or updated by the anonymizer service. The expiry date/time is calculated as described before, in the core service. The anonymizer parse the data\_status attribute value, converts the value to a real date and time, by adding the current time to . The anonymizer stores only one record for each user attempting to access a specified dataset. As shown in Table 5.3, the user ID=123456 has only one record for accessing data ID=111111. Another record for the same user can be created, if the user accessed another dataset. Therefore, each record will be created once and updated on every user’s access.

The purge process cannot create any record, instead it just updates the available data records. The last four attributes, in Table 5.3, are updated by the purge process. The purge process is an infinitive process, which is considered to be a part of the anonymizer service. The process is triggered as per time schedule. It can be setup for an execution as desired, for instance every 3 hours. The process reads all records of the log file, where {(purged=null/No), (Purge Result ≠ Fail), and (expiry date/time ≥ current date/time)}. The purge process deletes all expired copies, by the following the anonymization path for each copy. After the deletion completion, the process updated the data record by {last purge attempt, No. of Attempts, purge results and purged (Yes/No)}. However, some copies of datasets may be busy with other analytics operations or queries at the time of triggering the purge process. In this case, the purge process will wait for a period of time and skip few purge rounds before trying another attempt. For this reason, the log file keeps the latest date/time for purge attempt. This delay is essential to give enough time for the query process to complete the task. The queries triggered by users may lock the datasets, while the process is ongoing. Therefore, trying to purge this busy dataset on each round is inefficient. Purge process uses some parameters stored in a setup XML file, known as purge.xml. The XML file is shown in Figure 5.18.

The XML file allows system administrators to setup the waiting period before the next round starts. Figure 5.18 shows the default value of waiting period, which 30 is minutes, while the next attempt is given by 3 times. This means that the purge process will try to delete the datasets, if failed, the next attempt will be commenced after 2 hours. That is 3 attempts with 30 minutes waiting period between each attempt. Moreover, the process will try to delete the dataset 4 times, as configure in Figure 5.18. With each time, the log file is updated by incrementing the No. of Attempts, and updating the rest of the attributes. The attribute (Purge Result) is updated with the values success/fail after the fourth attempt. If the result value was fail, then an email will be sent to the system administrator’s email (admin@service\_provider.com), as shown in Figure 5.18.

It was explained that two different process can delete the expired datasets. Both processes are part of the anonymizer service, and they read from the same log file. The anonymizer may delete datasets, only if data status is remove. However, the purge process is the main process that carries over the deletion operations.



*Figure 5.18- Sample of purge.xml file*

## Improvement to MDSBA Security

Sweeney has descibed few attacks aagainst k-anonymity method, as discussed in Chapter 2/ section 2.3.5. however, MDSBA maybe a prone for different attacks. The previously mentioned attacks are not possible in MDSBA. This refers to the consistency of anonymity. In MDSBA, Q-IDs anonymization refers to the attribute probabilities. The anonymization always starts with the lowest probability attribute. Hence, amending queries does not switch the anonymization process to other Q-ID attributes. Also, the class attribute, in MDSBA, must gain the k-anonymity equivalency principle. However, other possible attacks against MDSBA can be summarised into two types of attacks; obvious guess and Across Groups Unique Identifiers (AGUI). These two attacks are descibed in the next two sub-sections.

### Obvious Guess

An adversary may be able to guess the sensitive attribute (class), if the Q-ID attributes are known to the adversary. This violation may appear when a group of equivalent records have one class value. In this case, increasing or decreasing the number of equivalent records does not affect the security level. For instance, if a group of patients have one value of the class ‘Diabetes=positive’ and they share the same race, age, and state, then the intruder can obviously guess the diabetes state of the patient. This attack is simply defined as:

**Definition 1***: Obvious Guess attack may appear in MDSBA if a group fully equivalent records consist of one class value.*

In the Obvious Guess, an adversary may easily guess the sensitive attribute (class), and without having to identify the record. For example, in Seer data, we may have a data bag shown in Table 5.4. In the described bag example, the diagnostic may contain one value only. This class demonstrates the Obvious Guess breach, which can be interpreted as; any black person lives in the county 125, and was diagnosed on 1997, with an age of 25 must have a positive histology cancer. The Obvious Guess occurs if the bag contains only one class value.

Table 5.4. *Obvious Guess* example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Y. DIAGNOSIS** | **COUNTY** | **RACE** | **AGE** | **DIAGNOSTIC** |
| 1997 | 125 | Black | 25 | Positive histology |
| 1997 | 125 | Black | 25 | Positive histology |
| 1997 | 125 | Black | 25 | Positive histology |

### Across Groups Unique Identifiers (AGUI)

The second possible attack against MDSBA is caused by splitting Q-ID attributes into small groups. This unique identifier’s security threat is lower than the uniqueness appearing in each Q-ID group. The attacker needs to know almost everything about the victim’s background. This is believed to be a background knowledge attack, and defined as:

**Definition 2**: *Across Groups Unique Identifiers (AGUI) is a unique record that appears post the anonymization. The record is not anonymized due to its equivalency with the other records. Hence, its uniqueness appears across multiple Q-ID groups and disappears on individual Q-ID groups.*

Based on this definition, we may notice that AGUI attack is possible if the adversary knows most of the personal attributes in multiple Q-ID groups. In this case, AGUI severity will not be as high as personal re-identification records that are targeted by k-anonymity. However, this low-security threat should be reduced to the minimal, to inhibit its impact on security violation. MDSBA does not guarantee AGUI prevention, but it supports the minimal appearance of such records. Dividing attributes into small Q-ID groups is essential to reduce the anonymization computation cost, and to participate in the granular access process.

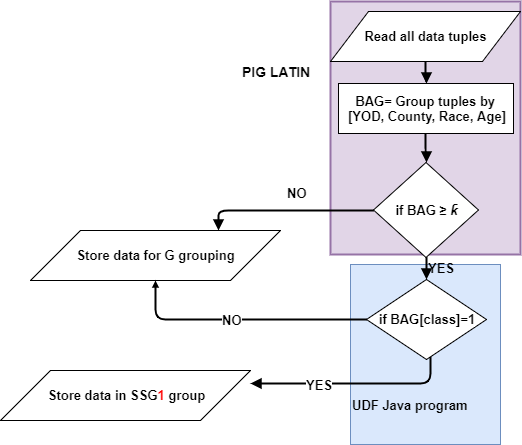
AGUI is caused by G(Q-ID) grouping, which may create another security concern. This may appear on dividing data attributes into small groups of Q-IDs and classes. Two to four Q-ID attributes are divided into groups with one class for each group. This grouping is essential to reduce the anonymization computation cost, and to participate in the granular access process. However, this grouping may help adversaries to spot out few known records. To explain this, let us take a look at Table 5.5 that shows a small part of the Seer data, with two Q-ID groups.

Table 5.5- Part of Seer data with two Q-ID groups

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Q-ID1** | | | | **Class-1** |  | **Q-ID2** | | | **Class-2** |
| **Age** | **Sex** | **County** | **State-county** | | **Year of diagnosis** | **Race** | **Grade** | **Diagnostic Confirmation** |
| 26 | Female | 31 | CT: Hartford | | 1991 | White | Unknown | Unknown |
| 26 | Female | 31 | NM: Chaves | | 1991 | White | Unknown | Unknown |
| 26 | Female | 31 | NM: Chaves | | 1991 | White | Unknown | Direct visualization |
| 26 | Female | 31 | WA: Jefferson | | 1991 | White | Unknown | Direct visualization |
| **26** | **Female** | **31** | **WA: Jefferson** | | **1991** | **Black** | **Moderately; Grade II** | **Positive histology** |
| 28 | Female | 31 | IA: Dubuque | | 1991 | Black | Moderately; Grade II | Positive histology |
| 28 | Female | 31 | UT: San Juan | | 1991 | Black | Moderately; Grade II | Clinical diagnosis |
| 28 | Female | 31 | UT: Utah | | 1991 | Black | Moderately; Grade II | Positive histology |
| 28 | Female | 31 | UT: Salt Lake | | 1991 | Black | B-cell; pre-B; B-precur. | Clinical Diagnosis |
| 28 | Female | 31 | WA: Jefferson | | 1991 | Black | B-cell; pre-B; B-precur. | Clinical Diagnosis o |

### Resolving Obvious Guess

To avoid Obvious Guess breach, initial filtration on stage zero can be implemented, and before splitting class values into groups. For the sake of performance, and to avoid the data overflow on Java Heap memory, this kind of filtration was implemented by using a simple UDF program. The program only checks the equivalent records of data bag that is ≥ k̄, if only one class value was found, then the data bag will be transferred to SSG group. This group aggregates the attributes based on the highest Q-ID propabilities. The lowest Q-ID propability will not be aggregated.in another word, the Obvious Guess data bag is considered as semi-equivalent records, even with the fully-equivalent data records. Single class value is prone to data leakage and can be noticed before anonymization starts. Figure 5.19 illustrates the zero stage procesure, which protects data bags from Obvious Guess records.



*Figure 5.19- Stage zero of anonymization to protect from Obvious Guess*

The UDF program reads the fully-equivalent data records that are greater than, or equal to k̄. The program loops only once for verifying the class values. If one value found, then the data bag is stored in SSG groups. This algorithm implements a small number of arrays, with one iteration process, which enhances the all over performance and increases the process scalability.

### Resolving Across Groups Unique Idntifier (AGUI)

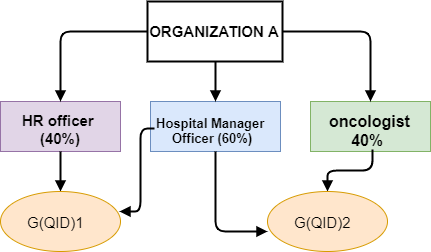
Resolving AGUI issue can be implemented by increasing the value of k for each Q-ID group. Therefore, assigning a quite large k value is highly recommended in MDSBA. The large k value leaves enough range of numbers to be assigned to the ownership levels, which provides the granular access method. Also, the large k value increases the anonymization percentage dramatically, which makes AGUI appears less frequent. Also, the large k value decreases the appearance of obvious guess records. It was experimentally proven, that obvious guess records may gradually decrease parallel with the k value increase, while data anonymity dramatically increases parallel with the k value increase. However, in most cases, the decrease of obvious guess is small if compared with the rapid increase of anonymized records. Therefore, a trade-off between anonymization loss and security threats reduction, should be considered. The value of k should be reasonable, which means neither too large nor too small.

Let us consider that Table 5.5 data has been anonymized with k(QID1)= 5, and k(QID2)=4. Each Q-ID group is anonymized separately. The highlighted record can be easily identified by adversaries, since the first five equivalent records of Q-ID1 overlap with the second four equivalent records of Q-ID2 [11]. Generally, the increase number of Q-IDs in any data may expose data to a higher rate of attack. This is because of the increase number of personal identifications or attributes. For instance, if knowing the patient’s age, gender, and postcode, may lead to uniquely identifying the patient with a probability of 87% [12], then the increase number of identifiers would increase the patient identifying even higher. However, Table 5.5 scenario possibility is low, because each Q-ID is anonymized individually, which allows a chance of randomness between both Q-ID groups. Nevertheless the scenario occurrence probability, there should be distinctive procedures to prevent or minimize the possibility of such occurrence.

## Experimenting Data Disruption in MDSBA Framework

In this chapter, MDSBA core structure was examined and tested with the defined services, and the security threats and resolutions. To evaluate MDSBA framework efficiency, a small lab was designed to demonstrate the three MDSBA services and the impact of security threats on them. The lab was setup at the University of Western Sydney as shown n Figure 5.7. The aim was to do an experiment about the framework components of the core, initializer and anonymizer services. The lab comprises four virtual machines (VM) and one laptop. The four VM are divided by one NameNode and two DataNodes for Hadoop domain. The third VM was setup as the SP-Gateway, and the forth VM was setup as an LDAP/Kerberos server. The laptop demonstrated the FS server, with a software application programmed in PHP language, with MySQL 5.6 database management system. Also, SAML server (Gluu Server [23]) was setup in both sides of FS laptop and SP-Gateway. SAML has successfully embedded the user’s attributes in the XML assertion file on transmitting attributes between both ends.

The experiment has adopted Seer dataset, with two G(QID) as shown in Table 5.5. An Organization named A was created and delegated to three different roles. The created roles are; HR Officer with = 40% (mapped to Q-ID1), Oncologist with = 40% (mapped to Q-ID2), and Hospital Manager with = 60% (mapped to both Q-ID groups). The organization and roles delegation percentages are illustrated in Figure 5.20. Also, three users were created with the following roles; user1 (HR Officer), user2 (Oncologist), and user3 (Hospital Manager).

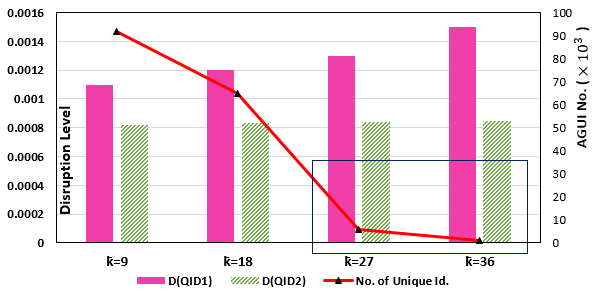


*Figure 5.20- Organization A and delegated roles*

The granular principle promotes a trade-off between the level of anonymized data and information gained. Fragmenting Q-IDs into groups to enhance the access control method is accepted technique if it does not contaminate the information gained predominantly. On the other hand, Q-ID groups may permit a security violation. The violation may occur with the multi-Q-ID access, while the anonymization is conducted on each Q-ID group separately. Some non-anonymized records are sufficient enough to create AGUI. These identifiers may occur as a result of anonymization randomness, which is conducted for each Q-ID group independently. The AGUI records appear when a user is permitted to access more than one Q-ID group. However, accessing more accessible Q-ID groups does not necessarily increase the number of AGUI records appearances. To reduce the AUGI in multiple Q-ID groups, an experiment should investigate the impact of increasing, which, theoretically, may reduce the percentage of AGUI records. However, this should be proven experimentally. Also, there is a need to find a compromise value for that can reduce the security violation and does not degrade the information usefulness at the same time.

The previously mentioned disruption equations were implements in these two experiments. The equations were detailed in Chapter 4 and formalized as: , where the anonymized blocks are calculated by finding the total summation of the disrupted blocks:. In this chapter experiments Seer Cancer data was used with some records N= 60,803,185 [24]. The data structure is shown in Table 5.5. Each Q-ID group was anonymized individually, by assigning random values of. Firstly, the three services were implemented and experimented regarding the automation of the anonymization process for different users. Next, the disruption values were calculated for each anonymized G(QID). Both Q-ID values were examined with similar set of values as follows; ={15,30,45,60}. All anonymized records, in both Q-IDs, were examined for AGUI records. Any data record that may re-identify a person with a unique record, and was not anonymized at all, is considered an AGUI record. The AGUI record uniqueness appears across multiple Q-IDs.

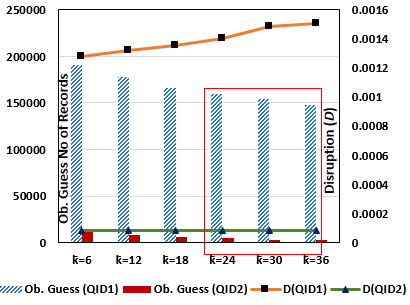
AGUI records were detected by transferring the anonymized data to SPSS program [25]. A small SPSS search script was applied to detect the AGUI records, by applying the uniqueness criterion. The criterion is finding any non-anonymized record with no equivalency with any other records. First, uer3 account was used, so data was anonymized based on = 60%. The number of AGUI was counted for each value of k-anonymity. Figure 5.21 compares between the D level, for both Q-ID groups, and the number of AGUI that appear in all records. The results show a slight increase in D on the larger values of for both Q-ID groups, but however, a dramatic decrease has appeared in AGUI number. This may indicate that AGUI number can be reduced by increasing the value of to a certain level. Certain level means finding a compromise solution that trade-off between the large number of disruption and the privacy violation. In Figure 4, the value of may output less anonymity loss in conjunction with the AGUI reduction impact. These results recall data owners to trade-off between D value and AGUI. This can be investigated by assigning large values of for roles that are permitted to access more than one Q-ID group.



*Figure 5.21- Tradeoff between D and AGUI No. in [].*

In the second experiment, the granular access and its impact was tested with the number of obvious guess records. This is essential to identify a proper scale of minimum and maximum values for k and values. In this experiment,user1 and user2 were used individually to access each Q-ID group. It is clear that a single group access permission will not face any AGUI problem. However, obvious guess remains an obstacle, even with a single Q-ID group. Moreover, the anonymized values increase in direct proportion to k value, they also, decrease in reverse proportion to obvious guess. Hence, one factor increases the D, while another factor decreases it. This is a logical explanation for the slight increase in disruption level, as shown in Figure 5.21.

Figure 5.22 shows that the lower values of have increased the number of obvious guess records. Therefore, choosing an optimal percentage of is essential to compromise a value between low disruption level and low number of obvious guess records. In Figure 5.22, the indicated bars, with a red square, show the optimal range, which is between of 24 – 36. The optimal value in this range can output the lowest values of D and obvious guess. Both experiments show that choosing the proper values of k and is the solution AGUI and obvious guess violations.



*Figure 5.22- Single Q-ID group access and obvious guess.*

## Summary

Big data is prone to multiple user’s access. Hence more fundamental structural shift toward the big data granular access is needed. Few techniques can be implemented to support access granularity. Aggregating Q-ID attributes into separate groups may support the granularity approach. Another technique is the variant numbers of equivalent records. This is implemented by assigning a gradual disparity for k-anonymity value. These two techniques are supported in the proposed MDSBA framework. The framework leverages the MapReduce performance and scalability. Its operational steps correspond to MapReduce ecosystem structure. In this chapter experiments, a practical demonstration was conducted to measure the user granular access by applying Hadoop ecosystem tools. The method of granularity is related to k-anonymity adjustment, and Q-ID grouping. Hence, users can be assigned to pre-created business roles that are given a maximum percentage of ownership level. We demonstrated permissions assignments in MDSBA. The permissions are managed by creating Q-ID groups, mapping business roles to groups and delegating roles to organizations. Users are given the access permissions by the delegated organizations.

This chapter demonstrated the three components of MDSBA framework, which are deployed between Federation Service and Service Providers. The three components, of core, initializer and anonymizer, have shown an automated solution for the granular access control of data anonymization. The granularity results have appeared on discriminately anonymized data. The contrast between anonymization levels was calculated by a mathematical equation given as a Disruption D. The experiments showed some security violation on assigning business roles mapped to more than one Q-ID group. The violation appeared across groups as a unique identifier (AGUI). Reducing the impact of this violation requires choosing optimal values of k and. It was proven that the small values of k may cause a large number of AGUI records, while large value of k may cause a high information loss on anonymization. Hence, we need to find a compromise value of k that tradeoff between AGUI violation and information loss. Next chapter will focus on finding the best practice on assigning k and values for a better security and performance operations in big data

[1] S. Narayanan, *Securing Hadoop*: Olton : Packt Publishing, 2013., 2013.

[2] O. Org, "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0," p. 46, 15 March 200 2005.

[3] Venkadesh.M and A. C. S. Dr, "Study on Improving Web Security using SAML Token," *International Journal of Computer Technology and Applications,* vol. 04, p. 917, 2013.

[4] J. Paschoud, "SHIBBOLETH AND SAML: AT LAST, A VIABLE GLOBAL STANDARD FOR RESOURCE ACCESS MANAGEMENT," *New Review of Information Networking,* vol. 10, pp. 147-160, 2004.

[5] C. M. Garcia and R. Abilio, "Systems Integration Using Web Services, REST and SOAP: A Practical Report," *Revista de Sistemas de Informação da FSMA,* pp. 34-41, 2017.

[6] O. Canovas, G. Lopez, and A. F. Gomez-Skarmeta, "A credential conversion service for SAML-based scenarios," *PUBLIC KEY INFRASTRUCTURE, PROCEEDINGS,* vol. 3093, pp. 297-305, 2004.

[7] J. B. Rosenberg, *Securing Web services with WS-Security : demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption*. Indianapolis, IN: Indianapolis, IN : SAMS, 2004.

[8] G. S. coding. (2015). *phpseclib: SSH2 Feature List*. Available: <http://phpseclib.sourceforge.net/ssh/intro.html>

[9] O. O’Malley, K. Zhang, S. Radia, R. Marti, and C. Harrell, "Hadoop security design," *Yahoo, Inc., Tech. Rep,* 2009.

[10] R. A. Napier, "Secure automation: achieving least privilege with SSH, Sudo and Setuid," in *18th Large Installation System Administration Conference*, 2004, pp. 203-212.

[11] L. Sweeney, "ANONYMITY: A MODEL FOR PROTECTING PRIVACY " *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 10, pp. 557-570, 2002.

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

[13] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional *k*-anonymity," in *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, 2006, pp. 25-25.

[14] A. Meyerson and R. Williams, "On the complexity of optimal K-anonymity," in *PODS '04*, C. Beeri, Ed., ed: ACM, 2004, pp. 223-228.

[15] G. Server. (2016). *Installation Guide for Gluu Server*. Available: https://gluu.org/docs/installation-guide/install/

[16] D. W. Aha. (1988). *Heart Disease Data Se*. Available: https://archive.ics.uci.edu/ml/datasets/Heart+Disease