# CHAPTER 2 – BACKGROUND AND CHALLENGES

## Big Data: Research Challenges

The recent advancements of technology have led to a data transmission to the cloud network. The portable hardware devices, such as tablets, smart phones, and laptops, have urged investors to adopt the cloud infrastructure as an adequate solution. Software applications, were especially designed for these new devices, are technically correlated with the cloud. Big data term was coined to cope with the advancement trend. Big data exhibits unique characteristics if compared with traditional data. The unique characters are summarized in 3Vs of Volume, Velocity, and Variety. To deal with a massive size of date, we need efficient mechanisms to store, retrieve and analyze a large size of data. Hence, Volume refers to the massive size of stored data, with capability to scale up the storage size. Velocity is related to the performance and efficiency in handling data transmission and process. The transmission time describes the time spent for collecting or storing data among storage nodes within the cluster or across clusters. Variety is related to the variety of the data type of structured and unstructured. These big data characteristics call for new system architectures of data acquisition, transmission, storage, and large-scale data processing mechanisms.

Big data structure can be decomposed into three main layers, infrastructure, computation, and application. The infrastructure consists of pool of hardware devices, and device management systems. The virtualization system is part of this layer. This also include all applications related to network management and security. The second layer is the computation layer, which is a middleware between the infrastructure and the application layers. This layers is divided into three divisions; integration, management, and programming models. The integration is related data distribution and aggregation to and from data nodes within the cluster. This is presented by the file system. Many free source file systems were developed during the last decade, such as; [Quantcast File System](http://www.linuxlinks.com/article/20130411160837132/QuantcastFileSystem.html), Hadoop Distributed File System (HDFS), Ceph [1], Lustre, GlusterFS, Google File System (GFS), and PVFS. These proposed file systems were essential to replace the traditional network file systems. Network File System (NFS) is inefficient to handle very large data across many nodes. Moreover Storage Area Network (SAN) file system can be scaled-up, but it is extremely expensive as a reason of it is dependency on fiber channel. The recent file systems were especially designed for big data. Most of them are provided with a parallel computation, and a map-reduce concept or a like. Also, they provide Portable Operating System Interface (POSIX). The network connection between the servers and the storage disks, such as NA and SAN, is not recommended in Hadoop domain. Instead, the Direct Access Connection (DAS) is used, which is SCSI, SATA, or SAS. Eventually, many similarities in structure and operations are available in most big data file systems. The second division of management is related to big database management systems such as; NoSQL, SQL, and file systems. Finally, the third division is the programming model, which combines the management and file system together, and facilitates the data analysis applications. MapReduce [13], Dryad [42], Pregel [43], and Dremel [44] exemplify programming models.

The third layer is the application layer. This layer connects the application interface with the second layer of programming model. Both programming model and interface infer various data analysis functions such as; queries, statistical and classification. The application layer exploits the MapReduce and parallel distributed tools to query statistical analyses, machine learning and precision, classification and other analytics needs. The three consecutive layers are related to each other’s. Choosing the infrastructure should be considered based upon the application needs and functions. Choosing the proper file system is linked to the chosen programming model. Divisions and layers must be preplanned before establishing the infrastructure. For instance, HDFS does not efficiently operate with any storage virtualization like RAID, instead, I/O data is handled by Just Bunch of Drives (JBOD). Therefore, data storages shuld be provided with HDFS compatibility for the best performance. Another example, if an organization has decided to adopt NAS file system, then all storage devices must be provided with Fiber Channels [2].

In big data, it is essential to keep the three layers under continuous monitoring. The intensive amount of data may overwhelm the management level as a reason of choosing the inappropriate layer of infrastructure, management or application. This is a very sensitive and accurate choice, since an extra delay of microseconds may be accumulated exponentially with the continuous data increase. However, the complexity of the three layers, with verities of choices may mislead data owners on choosing the optimal solution for their applications. For management model, there are many kinds of databases. Even though, there is no optimal database for all data types. This depends on the workload scenario, speed of read and write, and many other options. Researchers have proposed general comparisons between database performance on read, write, latency, durability, synchronous and asynchronous [3] [4].

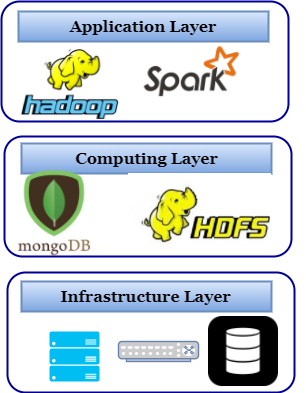


Figure 2.1- The three layers of big data structure

Researchers inferred that there is no single databases can make distinctive performance. Data of key-value, column, or document may perform differently with the varieties of database management systems. Also, database management models are different; some data operate on RAM, and keep a replication or a snapshot to disk. Other databases operate on disk and keep a cache copy to RAM. Also, some databases structure keep a high level of consistency and a low level of availability, or vice versa. Many NoSQL database are available in the market, MongDB and simpleDB for file data, Bigtable and HBase for Columns data, Dynamo and Redis for Key-Value, PNUTS for rows data. Eventually, choosing the proper database type is one of the challenges in big data.

In a similar concept to database management model, programming model also contains different options. Data owners need to choose between; batch processing model, graph processing model, stream processing model. The batch processing model deals with a snapshot of the targeted database. The stream processing model handles the real-time data, so the amount of copied data, from disk to RAM is very small. The third model is the graph processing model. This model suits some application, such as social media, where entities are related to one another. This model nature is iterative, and the same dataset is revised many times. The most popular graph processing model is the Pregel [43] and GraphLab.

Batch processing model consists of two user-defined functions, map and reduce, known as MapReduce operations. Their concept is expressed by performing data intensive computations in parallel distributed operations. A MapReduce reads input files from a distributed file system, which splits the data into multiple chunks. Each chunk is assigned to a mapper which reads the data, performs some computation, and outputs a list of key/value pairs. In the next phase, reducers combine the values belonging to each distinct key according to some functions and write the result into an output. The framework ensures fault-tolerant execution of mappers and reducers while scheduling them in parallel on any machine (node) in the system [5]. There are many batch processing models available in the IT industry, Hadoop and its ecosystems, and Dryad. Hadoop is a free-source framework developed by apache. Hadoop ecosystems resides at the top of Hadoop operations. An example of ecosystems are; Pig Latin, Hive, and Spark.

MapReduce models provide a range of varieties for business needs. Some MapReduce models operate in memory, while others operate in disk. Pig and Hive operate in disk, which reduces the efficiency of iterative and interactive jobs. This imposes a continuous reading from node’s disks on each MapReduce operation. Moreover, each set of iterative operations (query) is counted as a separate MapReduce job, which incurs a significant latency [6]. In spark, the concept is different, since it implements the resilient distributed dataset (RDD), which represents a read-only collection of objects partitioned across a set of machines. The RDD is explicitly cached in memory across nodes and reused in multiple MapReduce-like parallel operations. This creates a temporary copy of the data from the disk the RAM, so all iterative and interactive jobs are computed in RAM. This technique reduces latency, which is usually caused by travelling time spent on input and output with the disk.

The three big data layers include a large number of technologies and models. Designing an adequate big data network is not an easy task. The hard part is finding the most compatible design that suits the data type and structure. This makes it even harder when data contains verities of files, multimedia and database sets. This diversity may recursively appear in multi-tenants data structure. When some users have a massive size of files, while another user needs to deal with data stream project. This diversity has created a kind of complexity. These challenges urge researchers to find a proper platform that is able to deal with a dynamic stack of operations. The stack should be able to choose the best matrix performance based on the data type and structure.

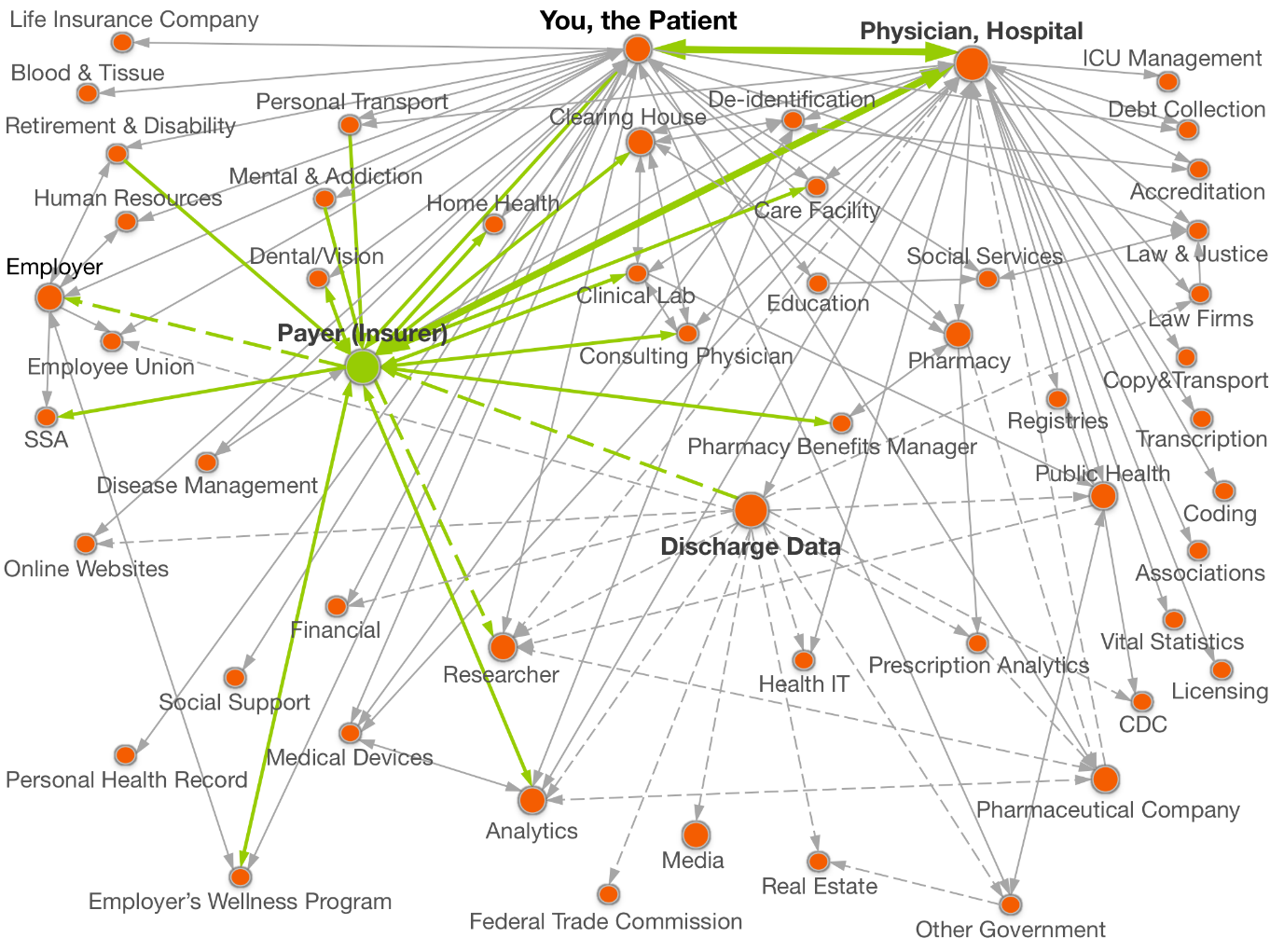
### Data Analytics

Data analytics is the most important part of big data. The aim is extracting useful information, which facilitates decision making, prediction, verifying the legitimate of data, or diagnosing and inferring faults reasons. The great diverse of data analytics methods and needs has derived several types of analytics criteria, the criteria can summarized in descriptive, predictive, and prescriptive. The descriptive analytics implements data mining for insight analysis to find what has happened in the past. The predictive analytics implements statistics and forecast methods to predict the future behavior. Eventually, the perspective analytics implements simulation to identify the system behavior, and as a result the decision making.

Big data analytics has gained more popularity as a reason of the new technology trends. These trends included new business applications that rely on data analytics, network applications, and scientific application evolution. The earliest business data were intuitive and simple. Relational data bases management systems were able to accommodate and operate the available amount of data. General stored data were structural. The Online Transaction Processing scaling (OLTP), and Online Analytical Process (OLAP) were operated in a small scale. Since the beginning of the new century, there was a large shift in data collection techniques. The Internet has supported companies to provide some of their data online, which gives their own customers more interaction with their business, and better automation for their systems. [7]. A tremendous amount of products and customer information were offered by online participants. The clickstream data logs provided companies with an opportunity to study customer’s behavior, and needed products and favorite services. Another wave of evolution has arisen after 2010, presented by smartphones. The number of sold smartphones and tablets exceeded the number of laptops and PCs. Portable devices and the Internet of Things created new features, such as location tracking, person-centered care, and context awareness.

The new evolution of smartphones supported the development of new services, as a reason of the increased number of users. Most of these services were not possible few years back. Moreover, smartphones, technically, are known by non-pc computers, which imposes limitations in processing and storing data. The amount of data produced by individuals exceeds the capacity of these devices. This concern moved user’s data to the cloud network. Currently, most mobiles applications run on a network. The majority global data is dominated by cloud and network. The largest data size occupied by users is multimedia data such as images and videos. Social has participated to a great extent in the data growth. It is estimated that more than 500 TB of data are uploaded on Facebook servers every day. Moreover, scientific research produce a huge volume of data from the fields of astrophysics and oceanography to genomics and environmental research. The National Science Foundation (NSF) has announced a BIGDATA project, which aims to advance the core scientific and technological means of managing, analyzing, visualizing, and extracting useful information from large, and heterogeneous data to accelerate the progress of scientific discovery and innovation.

Medical data is one of the prominent data that have an intensive analysis demand. Patient’s data is precious for many parties and organizations. Medical data is rotated around and distributed to many medical and non-medical organizations. It is difficult to trace the medical data since the mesh network of transmitting information contains more than 50 different departments. Figure 2.1 illustrated the complexity of tracing patient’s data, and the critical need for such data. The figure shows one example of Insurers how they are able to access the Physician, hospital, patient, pharmacy, work, and many other locations with and without the patient’s name. This high demand on medical data recall a need for establishing a complete data access framework, with a fine grained access privileges.



Legend: http://thedatamap.org/legend-solid.jpg with your name,http://thedatamap.org/legend-dashed.jpg without your name

Figure 2.1- Data map shows the patient’s information distribution -source [8]

## Big Data analytics Challenges

Analytics technique in big data is unlike the traditional data analytics. Some mathematical, statistical, prediction methods, and simulation are similar in both traditional and big data. However, the technique applied to calculate and conclude results is different. In big data, the un-structural data is mined and converted to structural data [9]. However, the volume of big data remains the main challenge in data analytics. Two major paradigms are expressed in big data analytics, batch and streaming analytics. Some applications may require fast and real-time analytics, while others not. Real-time analytics is needed in stock-trading analysis and alerts offered by financial services, fraud detection by examining transaction data, data and identity protection services, data generated by sensors and actuators embedded in physical objects, which is related to IoT, customer relationship management (CRM) applications, and clickstream analytics. In such applications, the processing time is essential, which shouldn’t exceed few milliseconds. The streaming process is continuous and infinitive, since the size of data is unknown. The infinite process fetches any new upcoming data, and proceeds this small portion to RAM. The fetching iteration is continuous, so the size of processed data is always small. The latest streaming frameworks are Storm, Flink, Kafka, and Spark [10].

In the batch-processing, data volume is known, the processing time is finite, and may last for seconds, minutes or even hours. Large data size is fetched from the storage, copied to RAM and processed. Therefore, large size of RAM and CPU is essential. MapReduce is the dominant model in batch processing. Data is divided into small chunks of data. The chunk are created by the file system and in parallel distributed manner in two phases, Map and reduce phases. This model schedules computation resources close to data location, which avoids the communication overhead of data transmission. The MapReduce model is widely applied in bioinformatics, web mining, census data, medical data, and machine learning. Depending on the application requirements, we may use streaming or batching mods. The differences in these two mods may encompass complex data storage and management systems, whereas in streaming mod, there is no data management system. One of the most popular MapReduce model is known by Hadoop. This model was especially designed for batch mode. Some models were especially designed to operate in the streaming mods, such as Storm, Samza, and Flink, while other models can operate in both mods, such as Spark.

Big data tools have gained a dramatic progress during the recent decade. Stream and batch tools are efficient enough to handle millions of data records within few seconds. Technologies like MapReduce have resolved the batch processing obstacles, while other technologies like Lambda architecture have resolved the stream and batch processing obstacles. Lambda technology consists of many frameworks such as Apache S4, Spark, Storm, Flink, and others. Some of these frameworks are dedicated for real-time and stream operations, while others can operate in both of stream and batch such as Spark. However, some operations in big data do not only need efficient frameworks, they also require efficient algorithms to benefit from the newly developed frameworks. One of the main parallel computing concerns is the algorithm structure. Ordinary programming and algorithms are inefficient in parallel processing. Special algorithms should be studied carefully to take the parallelism in to consideration. [11].

### MapReduce and Hadoop

MapReduce model is the dominator in batch processing framework. Hadoop can run MapReduce programs with the help of various languages such as; Java, Ruby, Python, and C++. Hadoop version 2 structure consists of two major divisions; Hadoop Distributed File System, and Yarn. MapReduce programs operate in parallel, so they can read and process a large data size at once. HDFS is a block structured distributed file system that is can store petabytes of data within multiple nodes. Each block of data is replicated on at least three different nodes within the cluster. MapReduce network consists of a NameNode and DataNodes. The NameNode is the master, while DataNodes are the slaves. NameNode stores, manages, and serves the metadata of the file system. Hence, NameNode does not keep the file system table, instead, it just recognized the data block availability and replication on which DataNode. DataNode manages and stores the actual data blocks as a per node service [12].

HDFS splits data into large chunk of blocks, 128 MB or larger. Therefore, the small size of files are not recommended in HDFS storage. DataNodes read data blocks from the HDFS, and transfer a copy of each block to parallel nodes for map processing. The reducer has three primary phases, shuffle, sort, and reduce. In the shuffle, the reducer copies the sorted output from each [Mapper](https://hadoop.apache.org/docs/r2.7.0/api/org/apache/hadoop/mapreduce/Mapper.html) using HTTP across the network. The sort phase sorts and merges the input by keys. The shuffle and sort phases occur simultaneously. HDFS operates with NoSQL database such as; HBase, and Casandra. Hive also supports the data management and collection as a warehouse. HDFS was developed especially for files management, therefore, HBase and other databases do not use all HDFS features and functions. For instance HBase is built at the top of HDFS, so HDFS process the data replication over nodes, by copying each block from the local server to the region server, which is a collection of data nodes. However, HBase is a dynamic database management system that enables many reads and writes features. Hence, HDFS is suitable for batch processes, while HBase is ideally suited for data stream [13].

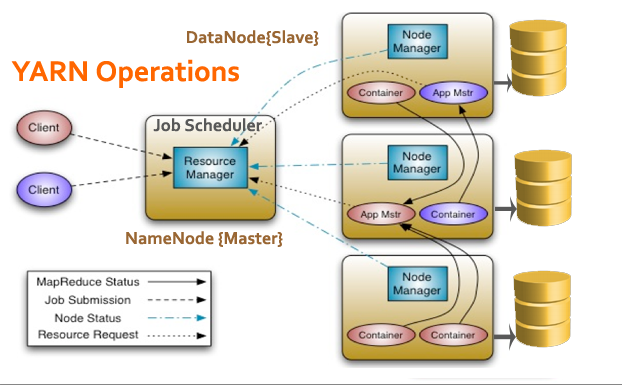


Figure 2.2-YARN structure in MapReduce

Yet Another Resource Negotiator (YARN) is the new feature introduced in Hadoop v2. YARN is the architectural core of Hadoop that allows multiple data processing such as; interactive SQL, streaming, and batch processing. YARN is the foundation of the new generation of Hadoop, which enable organizations everywhere to realize a modern data architecture. It allows multiple distributed processing frameworks to effectively share the resources of a Hadoop cluster, as shown in Figure 2.3

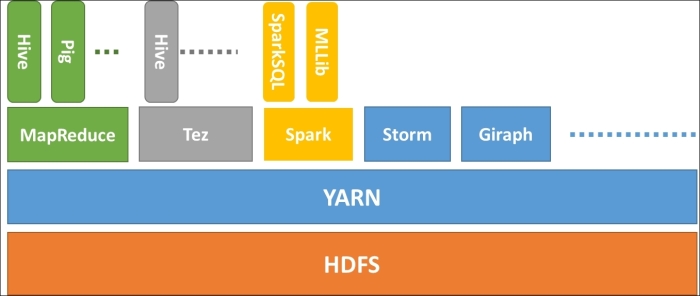


Figure 2.3- YARN supports an effective share of resources.

YARN manages nodes with the support of two main services, ResourceManager, and ApplicationMaster. The ResourceManager is interactively updated with the node resources and availability by the NodeManager. The node resources include: CPU, disk, and RAM. YARN is initiated when the user submits a job to Hive, Pig, or to any other models. ApplicationMaster is a per-application process that manages and coordinates the computations for a single application. Initially, ApplicationMaster is triggered by the user job submission. After job submission, the ApplicationMaster triggers the ResourceManager to obtain resources for the application. The ResourceManager deploys the NodeManager to find out the maximum available values of resource allocation. The availability is measured by a unit, called container. Each container of available resources has the right amount of CPU, disk, and RAM. Two main calculation operations are conducted by the NodeManager, NUM\_OF\_CONTAINERS, and RAM\_PER\_CONTAINER. The. This equation assigns the minimum value of the three resources to determine the number of containers. The. The reserved system memory is an essential term in both equations to calculate the total available RAM. Developers have suggested values of reserved RAM size as per total RAM size of each node. The suggested values are shown Table 2.1. Table 2.2 shows the recommended MIN\_CONATINER\_SIZE [14]. For instance, if the number of CPU cores in each node is 16, the number of disks in each node is 10, and the available memory is 48 GB, then NUM\_OF\_CONTAINERS =MIN(16\*2, 1.8\*10,(48-6)/2)=MIN(32,18,21)=18. The RAM\_PER\_CONTAINER=MAX(2,(48-6)/18)=MAX(2,2.3)=2.3≈2.

Once the ResourceManager defines the number of containers for the submitted job, the ApplicationMaster coordinates with the NodeManager to launch and monitor the application containers in the allocated resources. Leaving the coordination responsibilities to the ApplicationMaster will reduces the burden on the ResourceManager and will allow it to focus solely on managing the cluster resources. Moreover, creating an ApplicationMaster process for each separate submitted job improves the cluster scalability and performance.

Table 2.1. Reserved system memory

|  |  |
| --- | --- |
| **Total available RAM per Node** | **Reserved System Memory** |
| 4 GB | 1 GB |
| 8 GB | 2 GB |
| 24 GB | 4 GB |
| 48 GB | 6 GB |
| 64 GB | 8 GB |
| 256 GB | 32 GB |

Table 2.2. Minimum container size recommendations

|  |  |
| --- | --- |
| **Total RAM per Node** | **Recommended MIN\_CONTAINER\_SIZE** |
| Less than 4 GB | 256 MB |
| Between 4 GB and 8 GB | 512 MB |
| Between 8 GB and 24 GB | 1024 MB |
| Above 24 GB | 2048 MB |

### Batching and Pig

Varieties of frameworks were developed for big data. MapReduce is one of the strongest base framework in big data processing. However, MapReduce management tools were away from the traditional DBMS, hence, there was a need to develop advance tools that are able to mimic traditional data tools. In the first version of Hadoop, there was no supportive tools to deal with structural data, such as SQL. Pig data-flow was developed to narrow the gap between SQL and MapReduce. It is a high-level platform for creating MapReduce program. Pig offers SQL-like data modification constructs, which can be assembled in an explicit dataflow and interleaved with custom MapReduce style functions. Pig programs consist of a sequence of commands that are compiled into sequences of MapReduce jobs. Pig is an open-source project administered by the Apache Software Foundation. Pig compiles dataflow programs by a language called Pig Latin [15].

Traditionally, SQL language is the database dominator to manage and alter data. Hive is the data warehouse system for Hadoop, which aims to simplify Hadoop usage for data workers by providing the SQL-like language for Hadoop [16]. Pig Latin is another Hadoop tool that manages warehouse system, by using a proprietary scripting language. Pig Latin treats data as a set of tuples, which fosters tackling very large data sets. Thereby, substantial parallelism and a slew of optimization techniques are supported. Pig provides customized program for a User Defined Function (UDF), by supporting many common languages such as Java, Python [17], JavaScript, Ruby [18] or Groovy [19]. Similar to Hive, Pig supports ad-hoc queries, joins, and other SQL-like operations [20]. Pig compiler resides at the top of YARN and HDFS, which results in the client being responsible for running the script. Pig Latin is a combination notation of SQL-like and Java idiom. It allows three modes of user interaction; Interactive, batch, and embedded mode. In the interactive mode, the client is presented with an interactive shell, called Grunt. The interface allows user’s interactive commands line by line. This mode is suitable for developers and debuggers. The batch mode is the production mode, where users write a complete code and stores it in a file with an extension of (.pig). Finally, in the embedded mode, Pig is provided within Java library, by writing a Java code, and calling Pig library from inside Java [21].

Pig controls the data flow of tuples by creating data bags and maps. The bags are aggregated tuples of potentially varying structures, which may contain duplicates. Pig Latin was designed based on Functional Reactive Programming (FRP). The FRP depends on lazy evaluation and push/pull model. Pig was designed by pull or (iterator) model through the execution pipeline. The pull model was chosen over the push model for many considerations, such as UDF, and bags nested inside tuples [22]. Pig operates in a similar way to hadoop, which includes; map, sort, combine, shuffle, merge, and reduce. Data is aggregated by tuples to conclude bags, which leverages execution performance and speed. Pig Latin operator is triggered by either (DUMP) or (STORE), known by evaluation. Without a (STORE) command, the operator does not execute any task as a lazy evaluation. This has an advantage in the logical plan, since this may optimize the program structure by using Directed Acyclic Graphs (DAG) [23]. Moreover, if more than one STORE command in the script, then data is split and multiplexed, so they are processed in parallel. The SPLIT operation maintains a one tuple input buffer for each sub-flow or split [24].

One of the technical difficulties that may face developers of big data models is the JVM limited heap memory. JVM developers recommend a maximum of 25% memory allocation of the total RAM, which causes a memory waste and Java Heap memory errors [25]. In MapReduce structure, created JVM containers may consume the major size of the memory, which reduces the Java Heap memory errors. Therefore, it is more efficient to implement JVM through MapReduce, rather than using locally created JVMs. Since UDF adopts local JVMs, then it can be considered as a bottle nick in MapReduce. Avoiding a large data size flowing to UDF is essential to reduce error rates by Java Heap memory failure. However, this does not totally prevent operations failure. When using Pig, there is a possibility of passing a large size of data bag for materializing data to database format. In the usual case, Pig Latin is able to cope with the large size of bags, by implementing (combine) between tuples. However, memory overflow may appear in Pig due to materialization of large bags of tuples between and inside operators. In some cases, Pig needs to materialize large bags inside the pipeline for holistic bag computation. For this reason, another technique of spill arise to avoid the over-flow. This technique transfers some data rows to the disk in a temporary location. In few cases, the spill may fail to manage the large data flow, which causes an error of (out-of-memory) exemption. such an error can be avoided by increasing the JVM heap memory, or modifying the Pig Latin script, or even both actions [15].

Pig is still under development and immature solution. A consider number of concerns have not been resolved yet. Pig misses out an optimized storage structures like indexes and column groups. Also, Pig shell needs a considerable time to start, and to clean up jobs. It was developed merely for batch processing, therefore, it is not efficient for data stream. Pig Latin language is inefficient to complete some programming algorithms, such as iteration, nested iteration, and if statements. Therefore, developers need to implement many UDF programs. In the UDF, error messages are general and not clear. Developers agree that the biggest advantage of Pig is simplicity of coding Pig Latin script, and the smooth logic of data flow [26]. Unlike the coding complexity in Spark, as discussed next.

### Streaming and Spark

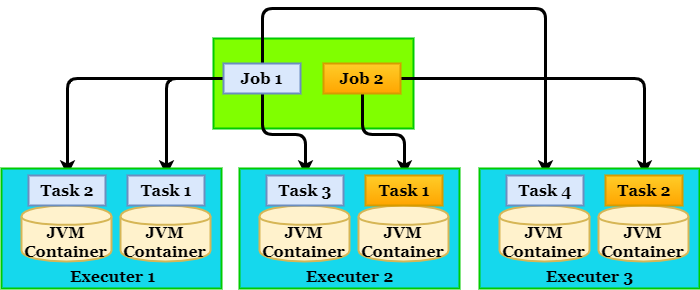


Figure 2.4- Spark structure and jobs distribution

Different tools were designed for each data type. Data streaming tools fetch the storage location continuously to read the newly updated data. This data is collected in small files and transmitted to the RAM for further processing. This process runs indefinitely, since the data size is unknown. The small size of data accelerates the processing speed, which is the aim of data streaming principle. On the other hand, data batching tools read all batched data only once, process them, and terminate after the process is completed. Streaming and batching tools vary in the core operation. Streaming need to access the storage units on a regular basis to read the most recent records, with a small size of collected batches or records. Hence, the operation latency should not cost more than few seconds or even milliseconds. Therefore, complex analytics cannot cope with the streaming principle [27].

Many recent frameworks were developed based on Lambda architecture, such as; Spark, Storm, Flink, Samza, and others. Most of these frameworks can be considered as hadoop ecosystems. This is because they are able to run at the top of YARN. One of the popular batching, and streaming frameworks is Spark. It has many advantages over the Pig Latin that mitigates the latencies, and increases the performance. Pig interacts intensively with the disk, by reading from HDFS and writing the results back to HDFS. These in/out transmissions consume a considerable time. Unlike Spark, which implements Resilient Distributed Dataset (RDD). RDDs is the main distinguishing feature of Spark. Spark adopts Scala as a de-facto language. The key programming abstraction in Spark is RDDs, which are fault-tolerant collections of objects partitioned across a cluster that can be manipulated in parallel. Users create RDDs by applying operations called transformations and actions. The actions triggers the lazy transformations. The transformations such as; map, filter, and groupBy cannot operate without proper actions such as; collect, count, and saveAsTextFile. Spark exposes RDDs through a functional programming API in Scala, Java, Python, and R, where users can simply pass local functions to run on the cluster. Spark may perform 2.5, 5, and 5 times faster than MapReduce [28].

Spark operations are different from the traditional MapReduce. Spark architecture is implemented to increase process performance. For this reason, multiple jobs can run in parallel by implementing; applications, executers, and active drivers. The traditional MapReduce splits each job into many tasks, and each task is processed by a single process within each container, so the process terminates when the task is completed. In Spark, each node contains one or more executers, and each executor operates in one container. The executor is a JVM base. The node may have many containers, which depends on the node capacity. Each container comprises one executor process that can run multiple tasks, and it remains for the life-time of the Spark life. This structure accelerates the initiation of the process and the tasks. Also, Spark consists of a process, known as active driver. This driver is used to manage the job flow and schedule tasks, and it is located on the master node. It interactively communicate with the executor of each node. If Spark was deployed on the top of YARN, then Spark driver can run over the cluster [29].

Spark is shipped with distribution processes that are able to mimic the functions of YARN. Users may install Spark in three different modes; stand-alone, cluster mode, and client mode. In stand-alone mode, the built-in resource manager is able to manage the cluster nodes, and without a need for YARN or Mesos. Both cluster and client modes are deployed over YARN. However, each mod was designed for different tasks. Users may run cluster mode for production jobs, while client mode is used for user interactive and debugging jobs. The main difference between both modes is the location of Spark driver. In the cluster mode, Spark driver runs inside Spark application master on the master node. This means that the user can prepare a script, for example Scala script, type the execution command, for instance spark-shell –i file.scala, and stay away. In the client mode, the driver is located inside the client process that initiates the Spark application. Therefore, users access the Spark shell, by using the command spark-shell”, and execute the script line by line [30].

One of Spark negatives is the programming difficulties that programmers may face. Spark operates in RAM, and programming with large data may derive Spark running out of heap memory. This is because of the unnecessary RDD data collection caused by the programmer algorithm. Programmers should have previous knowledge about Spark core structure and jobs like partitions, nodes, serialization, JVM, executors, memory and disk, shuffles, compressed files, and columnar formats (parquet). Therefore, they may need to try various algorithms to deduce the most efficient one. This frustrated and time consuming code may cause bugs within the program execution as soon as the data exceeds the maximum limit of resources. Usually, cached data that do not fit in memory are either spilled to disk or recomputed on the fly when needed, as determined by the RDD's storage level. However, this does not prevent data growth bugs and over-flow [31].

Anonymity in data analytics is an example of complex analytics, were anonymization operations scan the data records many times during the filtration, aggregation and masking operations. The anonymization processes latency is considerably high, therefore, batching tools are more efficient to deal with the large data size, and long latency. The big data tools were developed to accommodate both of data batches and streams. The first generations of MapReduce frameworks, such as hadoop, were unable to process the data stream. The next generation were developed based on Lambda architecture, which is designed to handle both batch and stream processing methods. The framework structure attempts to tradeoff between latency, throughput and fault-tolerance. Most of the, recently developed, real-time frameworks follow similar structure of storing temporary data frames and tables in the temporary random memory, so most of the operations are completed without performing input/output operations thereby decreasing latency [32].

Data analytics operations and the high latency are considered to be an obstacle in data anonymization. Therefore, data batching is currently the only option that is able to provide an intensive data analytics in big data. Analytics for data streams may provide more accurate results, but it is not possible for latency reason. Also, implementing k-anonymity cannot be implemented on data stream. The anonymity algorithm needs to identify the equivalent records for grouping them and masking the anonymity values. Therefore, we may choose a framework that is able to operate effetely in data batch.

## Security Challenges in Big Data Analytics

So far the previous sections addressed some challenges that face the analytics term in big. In this section, more security challenges will be addressed, with various solutions to resolve these concerns. Data analytics is prone for several attacks, and this can be categorized into three main attacks: storage, computation, and communication. In this research the main focus is the analytics operations in big data. Other kind of external or surrounding attacks are beyond the scope of this research. Miners, who attempt to access some datasets for analytics purposes, are prone to the three previously mentioned attacks [33]. Network administrator need to protect the network resources and operations of analytics.

Usually, big data are stored in multiple nodes and replicated on a number of 3X nodes. In MapReduce frameworks, data is either stored as files, or structured in a database management solution. It was previously mentioned that big data management tools are non-relational, and they contain a close structure to the file storage. Regardless the data format, data can be encrypted on multi-level, starting from the disk level to the data set level or even cell level. Big data encryption methods are very similar to the traditional data encryption, such as; transparent, column-level, field-level, file system, and hashing [33]. Unfortunately, storing data with such levels of encryption will degrade the performance as a reason of the high computation cost on decrypting data before being analyzed. Thereby, Homomorphic encryption was a proper solution, so miners can retrieve statistical results without unencrypting data. However, this type of encryption is immature yet, and still under research and development [34].

In computation and communication, MapReduce environment should be secure enough to handle miner’s analytics queries. Section 1.2 has already delved inside MapReduce framework. In Hadoop, the domain must be configured to switch to a secure mode. This includes processes and HDFS storages. Securing communication is directly related to securing computation as described in the next section. One of the major security challenges in big data analytics is the privacy re-identification attack. Medical, census, scientific, and commercial data may contain private details about individuals, who do not wish to share such information publically. The re-identification may occur even with hiding some attributes and values. In every dataset, there may be some sensitive information, such as medical status, should not be exposed to public. Researchers proposed different privacy protection techniques, by hiding all or part of the datasets. Each technique may best suit some research fields and needs in specific domains. Following, two privacy attacks and proposed methods for protections are presented.

The privacy re-identification may occur by several types of attacks. The attacks types are divided into three types; state attack, privacy attack, and timing attack. The state attack can be triggered by the adversary code, which may change the values of statistical variable, such as the keyword. In this case the privacy algorithms may lose the protection control. The attacker may run malicious code to transfer the other mapper’s output through the network. Another popular attack is the privacy, when the adversary reads some data and compares it with his/her external data. It is not necessary for the adversary to reach the sensitive data, it can be predicted based on the other attributes. The privacy attack may occur by side-link attack, or just guessing some private information as per homogeneity or background knowledge. Sometimes, users know some specific person’s private information, hence, guessing other sensitive information is not a hard task. Finally, the timing attack is possible by using an infinitive loop in the script, or by forcing scripts to run longer than the expected time. The time attack can also occur by the adversary using timing channel attack. The user keyword is also a prone for attacks [35] [36] [37].

### Protecting Privacy by Differential Privacy

The main challenge in big data analytics is the need for external users to access data. Data is given in interactive or non-interactive forms. The interactive forms mandate users to abstract statistical summary results without giving the actual data. This form conceals the actual data from users. Instead of showing data, attributes descriptions are given to allow users creating their own queries. In non-interactive forms, users are given anonymized data for security and privacy protection. The interactive form can be applied by encrypted or plain data. Users submit queries, the system completes the statistical calculations, and returns the statistical results to the user. However, this form of results provokes a kind of privacy attack. To illustrate the possible security breach, let us give the following example; Table 2.3 shows a list of patients, with attributes of age, gender, name, and diabetes status, where (Has Diabetes=1) means positive. If the miner (user) has submitted query Q1=”total number of Has Diabetes have diabetes”, while the second query Q2=”find the total number of Has Diabetes except Karen”. Since the user knows Karen’s name, and the abstraction of both queries Q2-Q1= 1, then Karen must have diabetes. This kind of attacks is very possible, even with hiding all data [35].

Table 2.3 Security attack using side information

|  |  |  |  |
| --- | --- | --- | --- |
| **Patient\_Age** | **Gender** | **Name** | **Has Diabetes** |
| 45 | Female | Marry | 1 |
| 40 | Male | Paul | 1 |
| 38 | Male | Mark | 0 |
| 55 | Female | Karen | 1 |
| 62 | Female | Nicole | 0 |
| 41 | Male | Steven | 1 |

Resolving such an attack is possible by applying differential privacy model [35, 36, 38, 39]. The model aims to eliminate some personal attacks, by adding noise to the input parameters or to the output results. The perturbation is a small numerical value that can be calculated by Laplace or Gaussian equations. Differential privacy are best defined as; the outcome of any analysis is, essentially equally likely, independent of whether any individual person joins or refrains from joining the dataset. In the previous example, a probability value can be added to the total number of patients with diabetes, so the results of the probability or noise value will be; P[have diabetes]=0.1, and P[have diabetes except Karen]=0.9 This protects the privacy re-identification of Karen, since the total of all patents have diabetes= 4.1, while the total of all patients have diabetes except Karen= 3.9. The privacy loss of each query is denoted by. This concept is mathematically stated as;

Differential Privacy was derived from two main thoughts. Firstly, the privacy method supposes that miners do not need to view data to retain any visible records. Secondly, increasing the statistical analytics queries may ruin the privacy and increase the re-identification probability. Hence, conducting deep analytics and creating extra analytics quires as per results output, may be contaminated by this privacy technique. In medical research, miners need to create queries as you go along with the output results. For instance, they may search for Anthrax symptoms in a specific region, while the output results show all symptoms similar to cold and flu. These results are inaccurate, and further queries are needed to filter out some common symptoms. Moreover, the added values of noise are generated automatically, which may leave a gap for queries manipulation by miners. One of the query manipulation method is choosing other known auxiliary data to mislead the system’s query recognition. Detecting queries manipulation depends on the algorithm used for detecting queries, before applying differential privacy [40].

### Differential Privacy Frameworks

Various differential privacy frameworks have been developed recently. The frameworks were specially developed for big data MapReduce operations. Software engineers still ongoing of developing differential privacy in real-life applications. Enterprise companies, such as Apple, have implemented differential privacy in their big data analytics. The most popular software applications are PINQ, Airavat, and GUPT.

#### Airavat

Airavat is a novel MapReduce security and privacy framework. It provides Mandatory Access Control MAC for Mappers, by enforcing MAC on both sides of MapReduce processes and Mapper output. Airavat follows the Analytics process step by step, starting from the user Mapper query, determining if the required keyword is single or multiple. MAC policy is enforced during the MapReduce processing. Finally, noise is added to the keyword, so the keyword can be compared with the key output. Airavat allows the execution of trusted and untrusted MapReduce on sensitive data, by enforcing the data owner policy using “Declassify Tag”. The latest versions of Airavat derived new security features such as; different side-channel attacks prevention, like state attack, privacy attack and time attack [41].

Airavat architecture divides the procedures between three parties: the data owner, the user or mapper, and the computation framework (Airavat). The user first plans his/her code for Map and Reduce. Two types of users are pre-defined in Airavat, trusted and untrusted, the untrusted user’s keyword is obfuscated by noise on output. Also, the untrusted user is prohibited from executing all queries, queries like “list all” is not permitted. In contrast, trusted user is permitted to use any queries. Airavat can’t confine the keyword as a sensitive value or not, since Airavat is unable to determine the string related attributes provided. Airavat uses differential privacy to create noise by using Laplace equations. Airavat sorts trusted user keys prior the output, so they don’t output in the same input sequence. As a result, the attacker can’t use the key order to leak information [36].

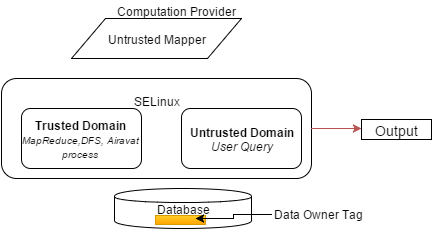


Figure 2.5- High-Level architecture of Airavat

SELinux is divided into two domains, one is untrusted for user code, and the other is trusted for: Airavat files, MapReduce, and DFS files. As shown in figure 2.5, both of these domains available in SELinux environment, the domain access and processes are controlled by MAC.

Airavat suffer from some limitations in confining the untrusted code, this is because of the keyword mapping difficulties, as described before. MAC is only used to control the user’s access and processes, while MAC doesn’t provide any mechanism for interacting with differential privacy or choosing sensitive attributes. The access control is not implemented to fully distinguish different levels of access, hence, the user is categorized as trusted or untrusted, which is two levels of privileges [42].

#### GUPT

GUPT was proposed to reduce the analytics complexity for an average programmer. GUPT considers that the data owner and the service provider are trusted, while the analyst is untrusted. To gain this aim, GUPT framework is divided into three blocks: data set manager, computation manager, and isolation execution chambers. Dataset manager is a database that maintains the data privacy, while computation manager handles the computations process by transferring data from the dataset manager, to the appropriate instances. Finally, the isolation execution chambers isolate and prevent any malicious behavior. GUPT uses the differential privacy to protect the final output. It also uses the Laplace to explore the noise level and accuracy. The analyzer query is evaluated on a smaller data blocks, since the search method implements small blocks of data and aging sensitivity. This means, the older age data are assigned by less sensitive values [36].

The optimal block size can be attained by finding a trade-off between the error and noise, which reduces the final error to a large extent. The block size varies from one query to another, this can be presented by , where *n* is the size of dataset, and is a parameter to be ascertained. Similar to Airavat, GUPT uses MAC to secure communication between instances. Each process performs in its designated space. AppArmor is used with SELinux as a sandbox to manage MAC [43]. The computation manager is split into server-client, the client allows GUPT to disable all network activates for the untrusted computation [36].

### Possible attacks in Differential Privacy

Differential privacy is prone to several covert-channel attacks, which includes state attack, time attack, and private budget attack. Several proposals have introduced some solutions for these attacks. Airavat and PINQ suggested two separate domains, where the trusted domain contains the actual database, and the untrusted domain is left for the untrusted users. The untrusted users initiate an untrusted query, and return the results through the network. The state and time attacks cannot be avoided if the user was able to reach the trusted domain, since these attacks require executing at least two programs, one is the query and the other is a program that may measure the execution time, the CPU speed, the power usage, and other parameters. The time attack depends on measuring the query time with and without one record. The time difference may determine the record status. The private budget attack may occur as a result of manipulating the differential privacy algorithm. The algorithm assigns equations to measure the user’s query budget. It can be expensive when it contains some private requests. For instance, when the query asks about specific person’s name, or personal details [37].

As explained earlier, most attacks channels are caused by the user’s query. Developers spend a considerable time to fetch the query and calculate the query costs. This is essential to apply the on the output results. Regardless, the strength of the differential privacy algorithm, it is impossible to secure all queries by predicting the actual cost for each query. There are unlimited number of queries that may manipulate or mislead the program. Moreover, queries can be initiated by more than one adversary. For instance, adversary 1 may initiate a query1, which contains all viewers who watch adult movies in X suburb, and there age fall between 30-40. If the number of viewers was little, then another user may obtain the adversary 1 one results to build another query accordingly. Adversaries can be a group of users who have certain agreements on querying from data. This may prove that preventing user’s queries and attacks is impossible, and there is always a change of security breaches [44].

### Protecting Privacy by K-Anonymity

It was previously mentioned that data analytics can be either interactive or non-interactive. It was also explained how interactive analytics can be protected by differential privacy method. In non-interactive analytics, data is modified and anonymized to thwart re-identification attacks by ensuring that no individual’s record is unique in the data. In non-interactive approach, miners are able to gain actual data view for analytics. This form gives more powerful tools to deeply analyze data with unlimited number of queries. The first anonymization method, known by k-anonymous, was proposed by Sweeney on 1998 [45]. More researchers have presented various methods related to k-anonymity concept.

One of the privacy techniques is the Quasi-Identifier (Q-ID), it implies finding a group of attributes that can identify other tuples in the database. These identifiers may not gain 100% of data, but even though, a risk of predicting some data remains high. For example, knowing the patient’s age, gender, and postcode, may lead to uniquely identifying that patient with 87%. Q-ID was implemented in k-anonymity method, and adopted as a scale for equivalency measurement. Only Q-ID attributes are verified for equivalency when investigating k value. However, Q-ID is a group of attributes chosen by data owners. So far, there is no clear technique to follow on assigning the Q-ID’s. Data owners choose a group of attributes, exposing these attributes together may thwart re-identification. Moreover, other auxiliary attributes may also support re-identification, therefore, Q-ID concept requires further study [46].

### Impairments in K-anonymity

As a new concept of privacy preservation, a considerable number of impairments were reported by researchers in regard to k-anonymity. It is essential to consider this on proposing any future work for privacy preserving frameworks. The following concerns were reported by a various number of studies:

* **Multiple queries and anonymity variations:**

Sweeney has addressed few possible security failure against k-anonymity method [47]. An adversary may submit multiple queries for analytics, then anonymization is applied on different Q-ID attributes on each anonymization query. Adversaries may request data several times with multi-queries, so the anonymization process may apply anonymity on the first Q-ID attribute in the first trial, and on the second Q-ID attribute in the second trial. Hence, linkage chance between the two anonymized tables is high. Table 2.4 illustrates an example of multiple-query attack.

* Table 2.4- Multiple anonymization example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GT1** | |  | **GT2** | |
| **Race** | **Zip** |  | **Race** | **Zip** |
| Person | 2138 | Asian | 2130 |
| Person | 2139 | Asian | 2130 |
| Person | 2139 | Black | 2140 |
| Person | 2138 | Black | 2140 |

The above example shows the impact of unsorted match between first and second anonymization. In GT1, the Q-ID attribute (Race) was anonymized, while in GT2, attribute Zip was anonymized. Hence, re-identifying such records is highly possible, by connecting GT1 with GT2, so both attributes can be easily identified. Table 2.4 reveals all actual values of Race and Zip by using two queries. In this case, k-anonymity masking is vain.

* Finding k-value in k-anonymity:

It was theoretically proven that finding k value is NP-hard [48] [49] [50]. Researchers studied the problem of anonymizing data by column suppression. They showed that this problem is NP-hard for k≥2. The complexity of this problem for k=2 remained open.

* **Curse of Dimensionality:**

The curse of dimensionality is described as the extra-ordinarily rapid growth in the dataset attributes as the number of personal identifications (or the dimension) increases. With the attributes increase, the cost of an algorithm grows exponentially with dimension, making the cost prohibitive for moderate or large values of the dimension [51]. Because of the recent increase of personal information amount, the curse of dimensionality becomes a real problem that may conclude a bottle neck on applying k-anonymity. A significant amount of work has been done on the privacy preservation concern of different types of data. Numerous models [52] [53] have been proposed for the privacy preservation. However, it has been proven that increasing the Q-ID attributes will make anonymization difficult with dimensionality increase [54] [55].

The high-dimensional attributes lead to a larger number personal attributes. Obviously, more personal attributes may increase the background attacks. The masking or perturbation added to one attribute need be increased parallel with the increase number of attributes. This imposes a higher amount of obfuscation and a lower information gained with the dimensionality increase. One of the suggested solutions to the curse of dimensionality is to find dependent personal attributes by implementing feature selection[56]. The feature selection can be used to determine the maximum dependent attributes, in order to reduce the dimensionality of the dataset and retain a small subset of attributes. However, reducing the dimensionality of datasets will negatively affect the significant amount of information gained. The feature selection process depends to attributes transformation, which inevitably impact the final statistical results. Another suggested solution to the curse of dimensionality is the concept of horizontal fragmentation. The idea is to break up the attributes into small subsets of attributes using horizontal fragmentation and anonymize each subset independently. The small anonymized subsets are then aggregated together. Even some attributes are anonymized, however, all attributes are retained. This method reduces the large information loss that may occur on applying feature selection method [57].

* **Background Knowledge:**

This kind of knowledge is one of the most manifested attacks. The attack may occur if the adversary has some background knowledge about the user, such as age, sex, address, nationality and others. For example, suppose that an adversary knows a Japanese man, of around 20’s of age was admitted to a specific hospital on that date. Table 2.5 shows the data list that the adversary have, where Table 2.5-A shows data before anonymization. The adversary is able to view Table 2.5-B only. When Alice has a pen-friend named Riku who is admitted to the same hospital as Bob, and whose patient record also appears in the table shown in Table 2.5-B. Alice knows that Umeko is a 21 year-old Japanese female who currently lives in the zip code 13068. Based on this information, Alice learns that Umeko’s information is contained in record number 1, 2, 3, or 4. Without additional information, Alice is not sure whether Umeko was diagnosed with HIV or heart disease. However, it is well-known that Japanese have an extremely low incidence of heart disease, especially in this young age. Therefore Alice concludes with near certainty that Umeko has an HIV [58].

Table 2.5-A Non-anonymized data sample Table 2.5-B Anonymized data sample

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Non-Sensitive** | | | **Class** |  | **Non-Sensitive** | | | **Class** |
| **Zip Code** | **Age** | **Nationality** | **Condition** |  | **Zip Code** | **Age** | **Nationality** | **Condition** |
| 13053 | 29 | Russian | Heart Disease |  | 130\*\* | <30 | \* | Heart Disease |
| 13068 | 28 | American | Heart Disease |  | 130\*\* | <30 | \* | Heart Disease |
| 13053 | 23 | Japanese | HIV |  | 130\*\* | <30 | \* | HIV |
| 13068 | 20 | Japanese | HIV |  | 130\*\* | <30 | \* | HIV |

* **Homogeneity Attack:**

This kind of attack may appear after completing the anonymization process. If a group of anonymized records contain similar sensitive information, then the adversary can obviously guess the person’s condition. Table 2.6 shows a sample of anonymized data, with similar sensitive values of (Cancer). If the victim lives in the post code 2116, then he must have cancer.

Table 2.6- Obvious Guess causes the Homogeneity attack.

|  |  |  |  |
| --- | --- | --- | --- |
| **Non-sensitive** | | | **Sensitive** |
| **Postcode** | **Age** | **Nationality** | **Condition** |
| 21\*\* | 3\* | \* | Cancer |
| 21\*\* | 3\* | \* | Cancer |
| 21\*\* | 3\* | \* | Cancer |
| 21\*\* | 3\* | \* | Cancer |

## K-Anonymity Frameworks

### K-Anonymity Methods for Traditional Data

#### Generalization

The generalization method was initially proposed by Sweeney. K-anonymity suggests a data generalization and suppression for quasi-identifiers (Q-ID). K-anonymity adopts the Q-ID definitions. It guarantees a privacy on releasing any record by adhering each record to at least k individuals, and this is correct even if the released records are connected to external information. The table is called k-anonymous; if one tuple has Q-ID values, and, at least, k – 1 equivalent records have Q-ID values. This means, the equivalence group size on QID is at least k [47]. The method is stated formally by defining any Q-ID table RT=(A1,…,An) , is said to be K-anonymity if each sequence of values in RT appears k times. The principle of this definition aggregates QIDs by domains. This implies attributes in the table TR, and each value in the table appears with a sequence of K occurrence [59].

The original k-anonymity method defines Minimum Generalization (MinGen), and Maximum Generalization (MaxGen). If the curator requests a query with two QID attributes, then the MinGen can be represented by omitting some values, or replacing them. The MaxGen implies values suppression, or hiding them completely. In each domain of a table T, a Domain Generalization Hierarchy DGH for an Attribute A is defined within a tuple t(A). The generalization g for the table T is defined as g(T). The generalization level (z) depends on the attribute value (νi). The following relationship implies the existence of the Value Generalization Hierarchy VGH for any attribute A for a function (f).

(2)

The generalization is defined as:

(3)

Some values can be generalized up to three levels before suppression has occurred like postcode, while other attributes are generalized to multiple z level, such as the taxonomy tree. The following examples illustrate the main concept of k-anonymity:

Postcode generalization DGH(Postcode): *Z0(2100,2109,2175), Z1(210\*,217\*), Z2(21\*\*), Z3(\*\*\*\*).*

Race Generalization DGH(Race): *Z0(Anglo,South American,African), Z1(person), Z2(\*\*\*\*\*\*).*

The generalized tables results are: GT(1,0), GT(1,1), GT(0,2), GT(0,1), as show in tables 2.6. Notice that GT(3,2), GT(2,2) and others are not possible in generalization, as they are assigned on suppression.

Table 2.7 Generalized tables GT

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Race (E0)** | **P.Code**  **(Z0)** |  | **Race (E1)** | **P.Code (Z0)** |  | **Race (E1)** | **P.Cod(Z1)** |  | **Race (E0)** | **Postcode**  **(Z2)** |  | **Race**  **(E0)** | **Postcode**  **(Z1)** |
| Anglo | 2100 |  | Person | 2100 |  | Person | 210\* |  | Anglo | 21\*\* |  | Anglo | 210\* |
| S.Amer | 2109 |  | Person | 2109 |  | Person | 210\* |  | S. Amer. | 21\*\* |  | S. Amer. | 210\* |
| African | 2100 |  | Person | 2100 |  | Person | 210\* |  | African | 21\*\* |  | African | 210\* |
| Anglo | 2175 |  | Person | 2175 |  | Person | 217\* |  | Anglo | 21\*\* |  | Anglo | 217\* |
| Anglo | 2109 |  | Person | 2109 |  | Person | 210\* |  | Anglo | 21\*\* |  | Anglo | 210\* |
| S.Amer | 2175 |  | Person | 2175 |  | Person | 217\* |  | S. Amer. | 21\*\* |  | S. Amer. | 217\* |
| African | 2100 |  | Person | 2100 |  | Person | 210\* |  | African | 21\*\* |  | African | 210\* |
| S.Amer | 2175 |  | Person | 2175 |  | Person | 217\* |  | S. Amer. | 21\*\* |  | S. Amer. | 217\* |

PT GT(1,0) GT(1,1) GT(0,2) GT(0,1)

The above tables can be distinguished by the precision value. The higher precision is the chosen generalization option. The precision table is calculated using the following equation:

In Table 2.7, DGH(Postcode)=3, and DGH(Race)=2 The number of attributes PT=2, and the number of tuples NA=8. Hence, the above table’s precision results are:

Prec(GT(1,0)=1- 8/2/16 = 0.75

Prec(GT(1,1)=1- (8/3 + 8/2)/16 = 0.58

Prec(GT(0,1)=1- 8/3/18 = 0.83

Prec(GT(0,2)=1- 16/3/16 = 0.67

The above calculated values prove that the highest precision is GT(0,1)=0.83, therefore, it will be picked up by the generalization algorithm [60].

The generalization using PT for each attribute is practically not possible for a large size of data, therefor, the real-world data is generalized and suppressed using tuples instead of individual attributes. One of these systems that can be implemented in the real-world data is datafly system [45]. The system guarantees the k-anonymity results, but does not necessarily guarantee the MinGen of data distortion. However, datafly is not very accurate, its decision is crude, since it generalizes all values associated with an attribute and suppresses all values within a tuple. Datafly is given the most important field, so it will be generalized, for example D\_O\_B is generalized to the year of birth instead. The next step is counting the number of times of the tuple occurrence. The non-repeated tuples with frequency=0 will be suppressed. Another popular system is μ-Argus, this system categorizes the attributes based on their sensitivity. The values given are: 0 (Not Identifying), 1 (Most Identifying), 2 (More Identifying), and 3 (Identifying) respectively. μ-Argus suppresses cells instead of suppressing the whole tuple, as mentioned in datafly [47].

The generalization algorithms apply similar concept in taxonomy trees and suppression. The concept endeavored to gain he k-anonymity by generalizing Q-ID attributes starting from the bottom of the tree, and moving upward at the top of the tree, known by Bottom-Up Generalization (BUG). Several algorithms and frameworks were proposed based on BUG concept. Here are two methods of Incognito, and *ℓ-diversity* as BUG examples.

##### Incognito

Incognito method is derived from *k-anonymity* model to limit the confidence of the implications from the quasi-identifier to a sensitive value. Incognito iterates the Q-ID attributes by generalizing the lattice nodes using join and prune. The method implements an algorithm of breadth-first search, by iterating the global-records several times to compute the frequency set of persons. SQL select queries are applied with (GROUP BY) for each two Q-IDs. The frequency number is calculated for each Q-ID attributes and compared with the other Q-ID attributes. The concept is finding Q-ID attributes that need to be anonymized before commencing the anonymization process. Finding them is conducted by calculating the most frequent appearance of each Q-ID and then building the lattice accordingly. The lattice is similar to the taxonomy tree concept. Each Q-ID is given two numbers, one denotes its sequence and other denotes the taxonomy level. For instance, the taxonomy tree of zero level for three Q-IDs is presented by:

*L0 (Level Zero) = {Q-ID(0,0), Q-ID(1,0), Q-ID(2,0)}*

After computing the frequency set for each root, the lattice results may impose two options of generalization. The first option moves one level up for the Q-ID0 taxonomy, one level up for the Q-ID1, and no generalization for Q-ID1, as shown below:

*L0 – L1= {Q-ID(0,1), Q-ID(1,1), Q-ID(2,0)} [first option]*

The second option moves one level up for Q-ID0, no generalization for Q-ID1, and two levels up for Q-ID2, as shown below:

*L0 –L1 – L2 = {Q-ID(0,1), Q-ID(1,0), Q-ID(2,2)} [second option]*

These value present the minimum anonymization level for non-equivalent records. This is an initial calculation for generalizing tuples globally. In the second calculation, the distance vector can be derived. The distance vector is calculated between two domain vectors {DA1… DAn} and {DB1... DBn} is a vector DV = [d1... dn], where each value di denotes the length of the path between the domain DAi and the domain DBi in the domain generalization hierarchy Hi. After calculating the distance victor, and determining the best lattice on generalization, a full-domain generalization is applied to all dataset [61].

This method gains better performance than the original BUG method of Sweeny and Samarati. The performance increases parallel with the increase number of Q-IDs, if compared with BUG. Table 2.8 compares the number of nodes searched between BUG and Incognito. However, several disclosures are inherited from the original k-anonymity method, such as the curse of dimensionality and background knowledge. Moreover, the full-domain generalization reduces the information gained in traditional datasets. Thus, Incognito leverages performance and reduces information usefulness.

Table 2.8- Comparison between Bottom-Up Generalization and Incognito methods

|  |  |  |
| --- | --- | --- |
| **Q-ID Size** | **Bottom-Up** | **Incognito** |
| 3 | 14 | 14 |
| 4 | 47 | 35 |
| 5 | 206 | 103 |
| 6 | 680 | 246 |
| 7 | 2088 | 664 |
| 8 | 6366 | 1778 |
| 9 | 12818 | 4307 |

##### *ℓ-diversity*

The *ℓ-diversity* is introduced by Machanavajjhala et al [62]. This algorithm aims to reduce the attributes linkage by preventing homogeneity attack and background knowledge. It is developed from the fact that a q block is *ℓ*-diverse if contains at least *ℓ* values for the sensitive attribute S.. The ℓ-diverse is calculated using the entropy, by grouping the Q-ID and calculating the entropy for the groups. Using the following:

(5)

For example, let us consider Table 2.9, as a part of age generalisation, and IQ={Patient\_Age,Race,Gender}, while the sensitive attribute is Disease. The records are grouped or compressed with the similar age, race, gender, and disease. Based on the above given definition, the entropy can be calculated as:

Table 2.9- illustrates *ℓ-diversity* model

|  |  |  |  |
| --- | --- | --- | --- |
| **Patient\_Age** | **Race** | **Gender** | **Disease** |
| 25-30 | Anglo | Male | Flu |
| 25-30 | Anglo | Male | Flu |
| 25-30 | Anglo | Male | Eczema |
| 30-35 | South American | Female | [Thalassemia](https://www.google.com.au/search?safe=off&biw=1242&bih=599&q=thalassemia&spell=1&sa=X&sqi=2&ved=0CBkQvwUoAGoVChMIsKn34oj-xwIVRuemCh3xdAXB) |
| 30-35 | South American | Female | [Thalassemia](https://www.google.com.au/search?safe=off&biw=1242&bih=599&q=thalassemia&spell=1&sa=X&sqi=2&ved=0CBkQvwUoAGoVChMIsKn34oj-xwIVRuemCh3xdAXB) |
| 30-35 | South American | Female | [Thalassemia](https://www.google.com.au/search?safe=off&biw=1242&bih=599&q=thalassemia&spell=1&sa=X&sqi=2&ved=0CBkQvwUoAGoVChMIsKn34oj-xwIVRuemCh3xdAXB) |
| 30-35 | South American | Female | Pneumonia |

The group <[25-30], Anglo, Male>

The group <[30-35],South American, Female>

To achieve entropy ℓ-diversity, the table as a whole must be at least log(ℓ), since the entropy of a Q-ID group is always greater than or equal to the minimum value. The minimum value of entropy ℓ -diversity =1.8, is considered to be the lowest value for the whole table [63].

The entropy ℓ-diversity is not effective in the real data environment, since grouping the similar records does not reduce the adversary attack possibility. In the above example, grouping the South American who have Thalassemia does not convey the possible successful percentage of 75%. The ℓ -diversity does not provide any measurement for portability-based risk.

#### Specialization

TDS algorithm was developed to achieve LKC-Privacy on high-dimensional data, the algorithm is also called HDTDS [64]. The idea is starting from the most general value in the taxonomy tree, and then move to the bottom of the tree. The taxonomy tree is pre-established for each attribute. At first, all tuples are generalized to the topmost root of the taxonomy tree (any), this value suppresses any quasi-identifier. The taxonomy tree is the prominent value that provides the masking operation in any attribute Dj. Masking operation can be identified by the data owner as a topmost tree parent node, and a child or a leaf nodes ν, written ν 🡪child (ν). A specialized Dj can be viewed as a cut of a tree denoted as cutj.

The taxonomy tree should be built in advance for each attribute. This technique implements the specialization for the Q-ID attributes as a masking method. Several methods were proposed based on TDS algorithm. However, all methods follow similar procedures in finding out the best score within the Q-ID attributes. The attribute with the best score will be specialized. The score equation is shown below.

(6)

The value 1 was added to avoid division by zero. This equation doesn’t satisfy the form matric to capture the classification, therefore, Shannon’s equation is used for correctness.

Next finding both of InfoGain and AnonyLoss to determine the best generalization for each attribute. This depends on the QID used on each analytics.

(7)

Where denotes the entropy of T(x). To find out the best score for a compressed table, let us consider Table 2.10 with an extra attribute of “Education”, which describes the patient’s education level starting from the year 9 to postgraduate studies. Herein the TDS model is used, and the compressed records must start from the root of taxonomy tree.

Table 2.10- Compressed Patient table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Education** | **Sex** | **Work\_hrs** | **Class** | **# of Records** |
| 10th | M | 40 | 20Y0N | 20 |
| 10th | M | 30 | 0Y4N | 4 |
| 9th | M | 30 | 0Y2N | 2 |
| 9th | F | 30 | 0Y4N | 4 |
| 9th | F | 40 | 0Y6N | 6 |
| 8th | F | 30 | 0Y2N | 2 |
| 8th | F | 40 | 0Y2N | 2 |
| **Total:** | | | **20Y20N** | **40** |

To calculate the InfoGain, and AnonyLoss for each the table, let us start first from the most top generalization, which is ANY\_Edu, for the whole records in the table.

* QID={Eudcation,sex,work\_hrs}
* Number of Records=40
* E(T[ANY\_Edu])=
* E(T[8th])=
* E(T[9th])=
* E(T[10th])=
* InfoGain(ANY\_Edu)= E(T[ANY\_Edu]) – (
* **InfoGain(ANY\_Edu)=1-(0+0+24/40\*0.65)=0.6**

While the InfoGain for the sex is calculated as:

* E(T[ANY\_Sex])=
* E(T[M])=
* E(T[F])=
* **InfoGain(ANY\_Sex)=E([ANY\_Sex])- (**
* E(T[1-99))=
* E(T[1-40))=
* E(T[40-99))=
* **InfoGain([1-99))=0.39**

Table 2.11, shows the specialization starts with Education, the highest InfoGain

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Education** | **Sex** | **Work\_hrs** | **Class** | **# of Records** |
| 10th | ANY\_Sex | [1-99) | 20Y4N | 24 |
| 9th | ANY\_Sex | [1-99) | 0Y12N | 12 |
| 8th | ANY\_Sex | [1-99) | 0Y4N | 4 |

The highest InfoGain is (ANY\_Edu), so the specialization for education starts first as shown in Table 2.11, in contrast to AnonyLoss, which shows “Sex” should be generalized first, as calculating AnonyLoss uses the following equation:

AnonyLoss (ANY\_Edu) = A (QID) – A (ANY\_Edu(QID))

The average of AnonyLoss is usually calculated to find out the best generalization and specialization for each attribute. The total results can be determine by calculating the score for each attribute, the score(v)=InfoGain / (InfoLoss + 1), for example, Score(ANY\_Edu)=0.0165, and Score(ANY\_Sex)=0.0183, and for [1-99)=0.0136. This can determines that the ANY\_Sex score is the highest.

##### LKC Privacy

LKC-privacy model can be applied for the multidimensional data, such as patient’s information. This method assumed that the original K-anonymity and its extended privacy models exaggerate the security risk, since they assume that an adversary could potentially use any of the Q-ID attributes as background knowledge attack. LKC-privacy assumes that it is very difficult for an adversary to acquire all personal details accurately to launch. Thus, it is assumes that the adversary’s background knowledge is bounded by at most L pairs of location and timestamp that the victim has visited. General intuition of LKC-privacy insures that Q-ID with a length of L and sensitive value of S is not greater than Class C, the idea is grouping length of records L in the data object T, by at least k records.

The following example illustrates the LKC-Privacy. Suppose that the following Table 2.12, and the taxonomy tree in Figure 2.6, where L=2, K=2, and C=50% (Yes or No). The Table 2.13 was generalized using Figure 2.6 taxonomy. Based on the given information, let us determine whether the generalization in table 2.13 is correct or not, in relate to LKC-Privacy model.

Table 2.12-, illustrates LKC-Privacy model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *Quasi-identifier (QID)* | | | *Class* | *Sensitive* |
| **ID** | **Job** | **Sex** | **Age** | **Transfuse** | **Surgery** |
| 1 | Cleaner | F | 35 | Y | Appendicitis |
| 2 | Cashier | F | 31 | Y | Appendicitis |
| 3 | Teacher | M | 35 | N | Urology |
| 4 | Engineer | M | 27 | N | Urology |
| 5 | Plumber | M | 25 | Y | Vascular |
| 6 | Electrician | M | 29 | N | Vascular |

Table 2.13, the previous table has been generalized for (Job, Age)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *Quasi-identifier (QID)* | | | *Class* | *Sensitive* |
| **ID** | **Job** | **Sex** | **Age** | **Transfuse** | **Surgery** |
| 1 | Non-Technical | F | 30-60 | Y | Appendicitis |
| 2 | Non-Technical | F | 30-60 | Y | Appendicitis |
| **3** | **Professional** | **M** | **30-60** | **N** | **Urology** |
| 4 | Professional | M | 1-30 | N | Urology |
| 5 | Technical | M | 1-30 | Y | Vascular |
| 6 | Technical | M | 1-30 | N | Vascular |

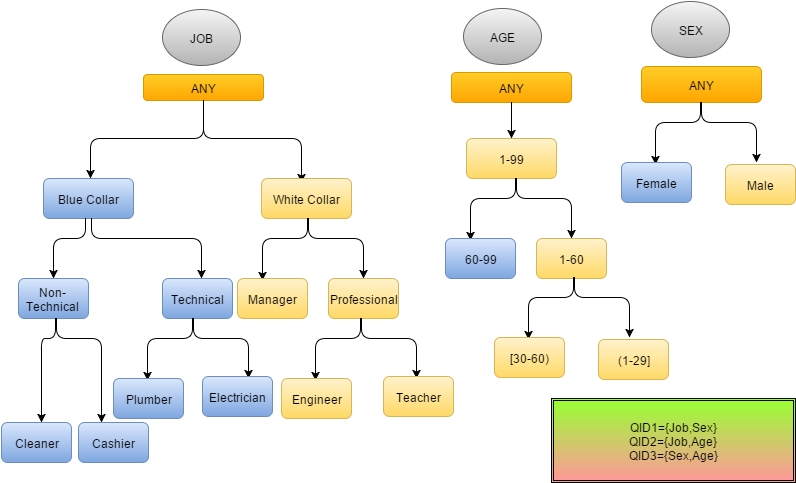


Figure 2.6- Taxonomy Trees for JOB, AGE, and SEX

As shown in Table 2.13, two records can be grouped together, so the generalization occurred in QID2 that supports the records grouping. Only record #3 of {professional, M, 30-60} cannot be grouped with the similar record, as the age interval is different. This implies another generalization level on age, for example between 0-90, which results higher utility loss [65].

This method can be suitable for dataset with larger number of k values. Small values of k may better anonymized by using BUG. Moreover, several privacy re-identification attacks cannot be resolved by using this method. The next (*α,k)-anonymization* is more effective*.*

##### (*α, k*)-anonymization

This anonymization method is categorized under TDS methods. It was introduced by Wong et al. 2006 [66]. The method aims to protect both identifications and sensitive associations in a disclosed dataset. Two approaches are developed to prevent discloser, one is an extra security applied on some chosen sensitive values but not all. The second approach is an extension of Incognito anonymization. The first approach assigns a decimal value *α* to some sensitive values in the attribute (Class). For instance, if users were diagnosed with HIV, then the HIV value must be exclusively protected more than the other sensitive values such as, Flu, or Headache

*(α,k)-anonymization* is an extension of Incognito with extra parameter α. The *α* denotes the –de-association requirement for the protection. The objective is to find local recoding with a minimum cost, or with a minimum number of suppressed records. Incognito algorithm is an optimal algorithm for the k-anonymity problem. It has many advantages over the other anonymization algorithms, such as resolving the diversity problem and making use of monotonicity property in searching the solution space. The search is a continuous iteration until finding the stopping condition. The stopping condition simply supposes that if a table T satisfies the privacy requirements, then every generalization of the table T also satisfies the privacy requirement. The extension of Incognito applies Top-Down Specialization (TDS) operations, which is an opposite technique to bottom-up Incognito.

The method proved that implemented TDS in Incognito reduces distortion by two to four time less. The generalization or BUG imposes the use of global recording on anonymization, which means one domain of generalized for the whole non-equivalent records in the dataset. The TDS use in Incognito is known by extended Incognito or (eIncognito). In eIncognito, the use of TDS has reduced the distortion as a reason of replacing the local recording by the global recording. The local recording is shown in Table 2.14-C. To understand eIncognito in depth, we need to understand the EDGE PARTITION INTO 4-CLIQUES in the graph theory [67]. Based on this theory, the anonymization cost is calculated for each record. For instance, if k=12, and α=0.5, we can interpret these two parameters by supposing that for 4 Q-IDs, there are 4 vertices in the 12 records corresponding to the edges in Q-IDs, then a cluster of these 12 records are formed where each modified record has four \*’s.

Let us study the following example to understand (α,k)-anonymization. Table 2.14-A shows a subset of dataset with three Q-IDs and one sensitive attribute. Some of the sensitive attributes are more sensitive than the others. For instance, people who are diagnosed with HIV should be strictly protected from re-identifying their status. The other diseases, such as Flu and Fever are not serious so there is no need to strictly hiding them. Assume that k=2 and α=0.5. Inspecting the anonymized Table 2.14-B carefully, we can see that the anonymized table does not protect two patients’ with sensitive information of HIV infection. We may easily distinguish the two patients for the first two tuples if we know that one of them live in the postcode 4350. Table 2.14-C is an appropriate solution. Since (\*,1975,4350) is linked to multiple diseases (i.e. HIV and fever) and (\*,\*,4350) is also linked to multiple diseases (i.e. HIV and flu), which protects individual identifications and hides the implication. Table 2.14-C shows the eIncognito anonymization using local recording. The table shows high InfoGain if compared with Table 2.14-D, which is anonymized by the old Incognito method. There are two goals for privacy preservation: (1) to protect individual identifications and (2) to protect sensitive relationships.

Table 2.14-A. Medical Data set Table 2.14-B. Anonymization pattern 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Job** | **Birth** | **Postcode** | **Illness** |  | **Job** | **Birth** | **Postcode** | **Illness** |
| Cat1 | 1975 | 4350 | HIV |  | Cat1 | \* | 4350 | HIV |
| Cat1 | 1955 | 4350 | HIV |  | Cat1 | \* | 4350 | HIV |
| Cat1 | 1955 | 5432 | Flu |  | Cat1 | 1955 | 5432 | Flu |
| Cat1 | 1955 | 5432 | Fever |  | Cat1 | 1955 | 5432 | Fever |
| Cat2 | 1975 | 4350 | Flu |  | Cat2 | 1975 | 4350 | Flu |
| Cat2 | 1975 | 4350 | Fever |  | Cat2 | 1975 | 4350 | Fever |
| Table 2.14-C. Anonymization pattern 2 | | | | Table 2.14-D. Global recording anonymization | | | | |
| **Job** | **Birth** | **Postcode** | **Illness** |  | **Job** | **Birth** | **Postcode** | **Illness** |
| \* | 1975 | 4350 | HIV |  | \* | \* | 4350 | HIV |
| \* | \* | 4350 | HIV |  | \* | \* | 4350 | HIV |
| Cat1 | 1955 | 5432 | Flu |  | \* | \* | 5432 | Flu |
| Cat1 | 1955 | 5432 | Fever |  | \* | \* | 5432 | Fever |
| \* | \* | 4350 | Flu |  | \* | \* | 4350 | Flu |
| \* | 1975 | 4350 | Fever |  | \* | \* | 4350 | Fever |

### Critic of Traditional Data Anonymization Methods

Traditional data is unlike big data. Traditional data comprises a limited number of data records. There is no threshold value to distinguish the traditional data from the big data. The rough data size and record’s numbers may provide a distinguisher between traditional and big data. Intuitively, we may consider few hundreds of thousands records are still traditional data. However, if anonymizing data can be accomplished by a single machine in an acceptable time-manner, and does not need a parallel distributed operations, then data can be considered as traditional data. Anonymizing traditional data algorithms do not require dataset spit into small blocks. Also, the limited number of records reduces the operations failure and errors. This is because the small size of data can be smoothly uploaded and fit in the current server’s memories. For these two reasons, there is no need to rectify the current known anonymization algorithms. The previously mentioned anonymization methods in section 2.4 are enough and can accomplish operations in an acceptable time period.

Choosing the best anonymization method is inaccurate. Some anonymization methods may be a reasonable choice for some datasets, but not for all. Different data records may require different anonymization methods. Two main types of anonymization methods can be chosen for various data types. In general, top-down specialization method is a suitable option when k value is quite large, while bottom-up generalization is a suitable choice when k value is small. Determining the k value may depend on the data divergence and type. Two main reasons may urge data owner to choose TDS algorithms over BUG; the large number of attribute values, and the organization’s security policy. More secure anonymized data impose larger k values. For instance, if data owner noticed that the attribute EDUCATION contains many values with a wide range of education varieties, then he/she may decide TDS algorithm. In another example, if the organization’s security policy is low, then the k value can be smaller than 4, so BUG algorithm is a good option.

### K-Anonymity Methods for Big Data

The previous BUG and TDS methods were also implemented in big data anonymization. Few amendments are applied to suit the big data frameworks, in the matter of parallelization and distribution. The core concept of k-anonymity is similar to the previously mentioned methods. Similar techniques and algorithms are applied in both cases of TDS and BUG. Let us study some of these anonymization methods to compare between the previously mentioned methods in traditional data and the big data.

#### Generalization

Several algorithms were proposed recently especially for anonymization using MapReduce. Most BUG methods follow similar algorithm, by implementing BUG driver to leverage the information gain and security trade-off. The search metric computes the Information Loss per Privacy Gain (ILPG). These equations measure the entropy and the scores of each attribute. The algorithm generates a random number (ran). This number presents the number of random partition for the dataset (). Each sub-dataset is emitted to the MRBUG driver for an intermediate generalization. This generalization scan data, find the equivalent records < k, and merge Q-IDs up to Anonymization Level one or two, that is AL1 or AL2. This intermediate generalization is essential to reduce the final anonymization computation. Finally, datasets are scanned again and to initialize a search metric ILPG. For each sub-dataset, if < k, then find the best generalization level and set to INACTIVE. Keep iterating and moving up the taxonomy tree, until k-anonymity is satisfied. As explained, the MRBUG driver operates twice, in intermediate and final. First, it merges anonymization, and second, it applies generalization. This algorithm is found in [68-70] [71].

##### Advanced BUG

Pandilakshmi et al. [70] proposed Advanced BUG (adv-BUG). The Advanced BUG consists of the following steps, data partition, run the MRBUG Driver on partitioned data set, combining the anonymization levels of the partitioned data set and applying generalization to original dataset [70]. Other anonymization methods use hybrid combination of BUG and TDS to anonymize data. A threshold value of k is determined by several algorithms to distinguish BUG from TDS use. The methods believe that BUG is more suitable for small k values, while TDS is more suitable for larger k values [72]. Some hybrid methods were recently proposed for big data by Zhang et al. and Irudayasamy et al. [68, 69, 73].

#### Specialization

Since the evolution of MapReduce and parallel processing, Roy et al. [41] presented a data privacy model, named Airavat . The system was developed after investigating MapReduce and differential privacy. This approach has encouraged researchers to re-design the available anonymization methods for MapReduce computability. The TDS methods for big data were derived from the TDS proposed for traditional data. The miner rectifies have been contributed to the early versions of MapReduce framework. One of the predominant methods is known by Two-Phase Top-Down Specialization (TPTDS).

##### Two-Phase Top-Down Specialization

TPTDS depicts the two phase of Map and Reduce. The concept is very similar to the previously explained TDS, which depends on generalizing all Q-ID attribute, and calculating the entropy and the score for each Q-ID. The highest Q-ID score will be specialized. This operation is iterated to find the best cut in the tree, or in the interval. In the first phase, dataset D is split into small chunk Di of data. Di denotes any block of data, where. The value denotes the number of blocks. An operation, known by MapReduce To-Down Specialization (*MRTDS)*, scans each data block in a subroutine in parallel to make full use of the job level parallelization of MapReduce. *MRTDS* driver is an intermediate anonymization level that specializes data without violating k –anonymity. *MRTDS* driver is applied once in each phases. In the first phase, the driver provides some sub-datasets a value, where. The term denotes the intermediate anonymity parameter, which is usually given by anonymization experts. Formally, the *MRTDS* operates multi-tasks on each data block for initial specialization by. The anonymization level presents the top generalization level of the taxonomy tree, which is usually given by (any). Specializing Q-ID attributes is applied as per highest score attribute. Another program, known by Information Gain per Privacy Loss (IGPL). The IGPL calculates the highest score for each specialized Q-ID attribute. This technique is popular in most anonymization operations and algorithms.

After completing the intermediate anonymization, all (AL) values are aggregated and the next phase is initiated. The second phase operates *MRTDS* again to produce the best cut specialization. The algorithm is close similar to phase one algorithm. The second phase receives data from the intermediate output as per key-value of (key,list(count)). This phase updates the IGPL results that was initiated in the first phase. Initially, phase one lists all best specialization for each data block. In the second phase, the specialization is validated or updated with a new specialization value. The validation is accomplished by gaining two conditions: firstly, the parent value of specialization should not be a root, i.e. should not be any. Secondly, the anonymity should be . Several iterations can find the best specialization cut for the chosen Q-ID. The IGPL updates the specialization list as per IG calculation, and the final list of specialization is updated and emitted, so the data records are masked with this list [74].

### Critic of Big Data Anonymization Methods

Most big data anonymization methods foster both TDS and BUG in a hybrid manner. The large data are managed and processed easier if hybrid technique was properly applied. This depends on the k value and other parameters. Both TDS and BUG were earlier used in traditional data, and there is no amendments have been done when implemented in big data. The concept of the Information Loss per Privacy Gain (ILPG) is the major core for all anonymized methods. This is true for traditional and big data. ILPG can be successfully implemented on a single machine, where ILPG driver iterates a large array of data records several times. The iteration attempts to find the best cut of generalization or specialization. In big data, this algorithm can be a cumbersome for memory size and processor limitations. Thus, amending ILPG algorithm to fit the distributed system can be gained by splitting dataset into small chunk or data. This is the exact solution adopted by the current anonymization methods. Chapter 3 proves that the data records equivalency increases parallel with the number of records increase. The positive fact of big data is the high percentage of records equivalency. Misusing this fact can negatively affect the information gain. We need a method that is able to benefit from this advantage, by splitting data in a nominal fashion, rather than conducting a random split. In the conclusion, the random split that is implemented in the current anonymization methods is inadequate.

In addition to the random split disadvantage, the current big data tools operate differently. ILPG driver was developed based on the early releases of MapReduce. The latest MapReduce and Spark frameworks are operated in two level-stack. The first level is the core structure of the distributed system attached with the second level of an ecosystem. For instance, MapReduce consists of many ecosystems operate at the top of YARN, such as Pig, Hive, and HBase. For this reason, implementing anonymization with the new released MapReduce ecosystems requires different algorithms. New released ecosystems diminished the iteration use, and limited the regular programming algorithms. For instance, Pig Latin script, Hive script, Scala script, and other programming scripts have limitations on using IF statements and iterations. These scripts were specially developed for parallel programming, and they are not flexible as the traditional programming. The reason is the scalability and performance concerns being considered on developing these programming scripts. If iterations and several IF statements are required, then user-define functions (UDF) can take a part in programming the needed part of the code. The UDF can play a small and limited part of the code. The UDF is a black-box, as explained in chapter 4. Data needs to flow outside the parallel system to a separate UDF. If the data flow was large, then there is no guarantee that the UDF will be able to handle this massive size of data. For this reason, data flowing to UDF must be limited and small. Unfortunately, ILPG algorithm depends completely on iteration and IF statements. The Algorithm was not rectified to fit the new ecosystems. Applying ILPG means converting the whole anonymization program to one UDF program. In another word, Parallel distributed framework will not be able to process ILPG efficiently, instead, data will be transferred and processed in a UDF, which is outside the parallel distributed framework.

So far, two critical concerns in the current anonymization methods. One more critical issue is regarding the scalability lack of the anonymization program. The current programs have restricted the number of Q-IDs. Maximum number of 9 Q-ID attributes can be assigned to each dataset. The increased number of Q-IDs may require an intensive computation cost. Imagine the ILPG needs to calculate the score for each attribute before deciding the specialized one. Hence, more Q-IDs will definitely reduce the speed and performance. We need an anonymization method that is able to use many Q-ID attributes. The need for increasing the number of Q-IDs and auxiliaries is high. The recent evolution of social media and portable hardware service have urged developers to increase the number of Q-IDs in multi-dimensional data. Adversaries can easily identify a person by searching the internet for some details about the person’s posts and a profile from Facebook, Twitter, Linked-In and others. We need an anonymization framework that is able to deal with many Q-ID attributes efficiently.

Increasing the number of Q-IDs in multi-dimensional dataset supports security and granularity. The increase demand on data analytics imposes better tools to deal with the authorization level and access control. More demands on data analytics means more users requesting access to data. The current anonymization methods cannot be consider as granular access control methods. The anonymization is applied evenly to all users. There is no any gradual access control for multiple users. The advancement of access control techniques exploits gradual and fine-access control to improve the security level. Since data owners may share data with business partners, strategic partners, co-owners, contractors, and public. User’s business background can determine the security level for each individual organization or user. The granularity can be applied on the level of anonymization. Hence, the user with a high security access is prone to a high level of anonymization, and vice versa.

## Summary

This chapter introduced the background of big data analytics and the challenges that face the big data analytics security at the present and in the near future. The chapter initially introduced big data framework structure, and the recent tools of managing such a massive size of data. Three layers of big data structure are introduced; infrastructure, computation, and application. The infrastructure refers to the hardware equipment in the data centers. The computation refers to the middleware of file systems on managing files and NoSQL on managing the big database. The application layer presents the parallel distributed frameworks that process queries of data analytics and cluster operations. The scope of this research focuses on data analytics security, therefore, data analytics was introduced by manifesting its importance in big data, and the challenges that face analytics. One of the analytics challenges is the processing performance and speed. Data analytics algorithms tend to operate intensive computations probabilities and statistics. Analytics tools, such as MapReduce, still have limitations in accomplishing such large jobs in a real-time manner. This is the major concern in big data analytics. Parallel computing frameworks conclude some tools were specially designed for batch processing, while other frameworks were designed for batch and stream processing. However, streaming tools are unable to operate efficiently in big data analytics. Two frameworks were introduced for each processing type, Pig for batch processing and Spark for stream processing.

The next following sections delve in data analytics and security. The sections review both of differential privacy and k-anonymity. Several methods support the differential privacy methods such as, Airavat and GUPT. Other methods support k-anonymity in traditional data and big data. The major difference between differential privacy and k-anonymity is the interactivity for each type. Differential privacy is an interactive form, while k-anonymity is a non-interactive form. The interactive form exploits statistical results without revealing data, in contrast to non-interactive form, which permits a partial data-access after anonymizing some data values. One major challenge in differential privacy is the user’s queries. Applying some obfuscation to the statistical results requires a full understanding to user’s queries. Some adversaries may intentionally create risky queries, so they can avoid the obfuscation. Researchers spent a considerable time on predicting and finding proper solutions for attacker’s queries.

Two techniques of k-anonymization were explained; generalization and specialization. The technique of generalization is opposite to the specialization. The generalization implements the taxonomy tree to move from the bottom of the tree up to the root of the tree. Therefore, it is known by Bottom-Up Generalization or BUG. The specialization implements the taxonomy tree to move from the top of the tree to the bottom tree. Therefore, it is known by Top-Down Specialization or TDS. Both BUG and TDS algorithms operate efficiently in traditional data. However, big data anonymization by TDS or BUG is inefficient for several reasons such as: data is split randomly, current anonymization methods do not cope with the latest technology of parallel distributed operations, and the scalability lack of the anonymization program.

[1] S. Weil, S. Brandt, E. Miller, D. Long, and C. Maltzahn, "Ceph: a scalable, high-performance distributed file system," ed, 2006, pp. 307-320.

[2] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE access,* vol. 2, pp. 652-687, 2014.

[3] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proceedings of the 1st ACM symposium on Cloud computing*, 2010, pp. 143-154.

[4] K. Keeton, C. B. Morrey III, C. A. Soules, and A. Veitch, "Lazybase: freshness vs. performance in information management," *ACM SIGOPS Operating Systems Review,* vol. 44, pp. 15-19, 2010.

[5] Q. Tran and H. Sato, "A solution for privacy protection in mapreduce," ed, 2012, pp. 515-520.

[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," *HotCloud,* vol. 10, p. 95, 2010.

[7] J. Richardson, K. Schlegel, B. Hostmann, and N. McMurchy, "Magic quadrant for business intelligence platforms," *Core research note G,* vol. 154227, 2008.

[8] L. S. J. Z. J. S. Y. S. Hooley. (2016). *The data map*. Available: https://thedatamap.org/map2013/p17.php

[9] S. Amini, I. Gerostathopoulos, and C. Prehofer, "Big data analytics architecture for real-time traffic control," in *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2017 5th IEEE International Conference on*, 2017, pp. 710-715.

[10] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management,* vol. 35, pp. 137-144, 2015.

[11] A. Bifet, "Mining Big Data in Real Time," *Informatica,* vol. 37, pp. 15-20, 2013.

[12] T. White, *Hadoop The Definitive Guide*, 4th ed. ed. Sebastopol: Sebastopol : O'Reilly Media, 2015.

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

[14] Hortonworks. (2016). *Determine YARN and MapReduce Memory Configuration Settings*. Available: https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.0.6.0/bk\_installing\_manually\_book/content/rpm-chap1-11.html

[15] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston*, et al.*, "Building a high-level dataflow system on top of Map-Reduce: the Pig experience," *Proceedings of the VLDB Endowment,* vol. 2, pp. 1414-1425, 2009.

[16] T. Dokeroglu, S. Ozal, M. A. Bayir, M. S. Cinar, and A. Cosar, "Improving the performance of Hadoop Hive by sharing scan and computation tasks," *Journal of Cloud Computing,* vol. 3, p. 1, 2014.

[17] M. Lutz, *Programming python* vol. 8: O'Reilly, 1996.

[18] M. Grimmer, "High-performance language interoperability in multi-language runtimes," in *Proceedings of the companion publication of the 2014 ACM SIGPLAN conference on Systems, Programming, and Applications: Software for Humanity*, 2014, pp. 17-19.

[19] D. Koenig, A. Glover, P. King, G. Laforge, and J. Skeet, *Groovy in action* vol. 1: Manning, 2007.

[20] B. Lublinsky, *Professional Hadoop Solutions*. Hoboken: Hoboken : Wiley, 2013.

[21] A. Gates and D. Dai, *Programming Pig: Dataflow Scripting with Hadoop*: " O'Reilly Media, Inc.", 2016.

[22] G. a. R. Peter. (2010, 23 Jan 2018). *Pig*. Available: https://www2.cs.duke.edu/courses/cps296.1/spring10/lectures/21-pig.pdf

[23] G. P. Gibilisco, M. Li, L. Zhang, and D. Ardagna, "Stage aware performance modeling of DAG based in memory analytic platforms," in *Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on*, 2016, pp. 188-195.

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

[25] K. Elmeleegy, C. Olston, and B. Reed, "Spongefiles: Mitigating data skew in mapreduce using distributed memory," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 551-562.

[26] V. Kalavri, V. Vlassov, and P. Brand, "Ponic: Using stratosphere to speed up pig analytics," in *European Conference on Parallel Processing*, 2013, pp. 279-290.

[27] J. Du, X. Gu, and T. Yu, "On verifying stateful dataflow processing services in large-scale cloud systems," in *CCS '10*, E. Al-shaer, A. D. Keromytis, and V. Shmatikov, Eds., ed: ACM, 2010, pp. 672-674.

[28] M. A. Abbasi, *Learning Apache Spark 2*, 1st ed. ed.: Birmingham : Packt Publishing, 2017., 2017.

[29] A. G. Shoro and T. R. Soomro, "Big data analysis: Apache spark perspective," *Global Journal of Computer Science and Technology,* vol. 15, 2015.

[30] S. Gopalani and R. Arora, "Comparing apache spark and map reduce with performance analysis using K-means," *International Journal of Computer Applications,* vol. 113, 2015.

[31] M. Frampton, *Mastering Apache Spark*: Packt Publishing Ltd, 2015.

[32] M. Guller, *Big Data Analytics with Spark A Practitioner's Guide to Using Spark for Large Scale Data Analysis*: Berkeley, CA : Apress : Imprint: Apress, 2015., 2015.

[33] Y. Gahi, M. Guennoun, and H. T. Mouftah, "Big Data Analytics: Security and privacy challenges," vol. 2016-, ed, 2016, pp. 952-957.

[34] M. Ogburn, C. Turner, and P. Dahal, "Homomorphic Encryption," *Procedia Computer Science,* vol. 20, pp. 502-509, 2013.

[35] A. R. Cynthia Dwork, "The Algorithmic Foundations of Differential Privacy," *Theoretical Computer Science,* vol. 9, 2014.

[36] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler, "GUPT: privacy preserving data analysis made easy," in *SIGMOD '12*, K. S. Candan, uk, Y. Chen, R. Snodgrass, L. Gravano, and A. Fuxman, Eds., ed: ACM, 2012, pp. 349-360.

[37] A. Haeberlen, B. C. Pierce, and A. Narayan, "Differential Privacy Under Fire," in *USENIX Security Symposium*, 2011.

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

[39] A. Roth and T. Roughgarden, "Interactive privacy via the median mechanism," ed, 2010, pp. 765-774.

[40] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, 2010, pp. 51-60.

[41] S. T. S. Indrajit Roy, Ann Kilzer, Vitaly Shmatikov, Emmett Witchel, "Airavat: Security and Privacy for MapReduce," *The University of Texas at Austin,* 2010.

[42] M. Nassar, A. A.-R. Orabi, M. Doha, and B. Al Bouna, "An SQL-like query tool for data anonymization and outsourcing," in *Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), 2015 International Conference on*, 2015, pp. 1-3.

[43] Z. C. Schreuders, T. J. McGill, and C. Payne, "Towards usable application-oriented access controls: qualitative results from a usability study of SELinux, AppArmor and FBAC-LSM," *International Journal of Information Security and Privacy,* vol. 6, pp. 57-76, 2012.

[44] N. Li, W. Qardaji, and D. Su, "On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, 2012, pp. 32-33.

[45] L. Sweeney, "Datafly: A system for providing anonymity in medical data," in *Database Security XI*, ed: Springer, 1998, pp. 356-381.

[46] R. Motwani and Y. Xu, "Efficient algorithms for masking and finding quasi-identifiers," in *Proceedings of the Conference on Very Large Data Bases (VLDB)*, 2007, pp. 83-93.

[47] L. Sweeney, "ACHIEVING -ANONYMITY PRIVACY PROTECTION USING GENERALIZATION AND SUPPRESSION," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 10, pp. 571-588, 2002.

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

[49] R. J. Bayardo and R. J. Rakesh Agrawal, "Data privacy through optimal k-anonymization," ed. USA, 2005, pp. 217-228.

[50] A. Scott, V. Srinivasan, and U. Stege, "k-Attribute-Anonymity is hard even for k= 2," *Information Processing Letters,* vol. 115, pp. 368-370, 2015.

[51] F. Y. Kuo and I. H. Sloan, "Lifting the curse of dimensionality," *Notices of the AMS,* vol. 52, pp. 1320-1328, 2005.

[52] S. Agrawal and J. R. Haritsa, "A framework for high-accuracy privacy-preserving mining," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, 2005, pp. 193-204.

[53] F. Li, J. Sun, S. Papadimitriou, G. A. Mihaila, and I. Stanoi, "Hiding in the crowd: Privacy preservation on evolving streams through correlation tracking," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, 2007, pp. 686-695.

[54] C. C. Aggarwal, "On k-anonymity and the curse of dimensionality," in *Proceedings of the 31st international conference on Very large data bases*, 2005, pp. 901-909.

[55] C. C. Aggarwal, "Privacy and the dimensionality curse," *Privacy-Preserving Data Mining,* pp. 433-460, 2008.

[56] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence,* vol. 27, pp. 1226-1238, 2005.

[57] H. Zakerzadeh, C. Aggarwal, and K. Barker, "Managing dimensionality in data privacy anonymization," *An International Journal,* vol. 49, pp. 341-373, 2016.

[58] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD),* vol. 1, pp. 3-es, 2007.

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

[60] B. C. M. Fung, K. Wang, and P. S. Yu, "Anonymizing Classification Data for Privacy Preservation," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 19, 2007.

[61] K. Lefevre, D. Dewitt, and R. Ramakrishnan, "Incognito: efficient full-domain K-anonymity," ed, 2005, pp. 49-60.

[62] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber, "Privacy: Theory meets Practice on the Map," ed, 2008, pp. 277-286.

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

[64] B. C. M. Fung, K. Wang, and P. S. Yu, "Top-down specialization for information and privacy preservation," ed. USA, 2005, pp. 205-216.

[65] N. Mohammed, B. Fung, and M. Debbabi, "Walking in the crowd: anonymizing trajectory data for pattern analysis," ed, 2009, pp. 1441-1444.

[66] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang, "(α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 754-759.

[67] J. Pardalos and M. Resende, "On maximum clique problems in very large graphs," *DIMACS series,* vol. 50, pp. 119-130, 1999.

[68] A. Irudayasamy and L. Arockiam, "Parallel bottom-up generalization approach for data anonymization using map reduce for security of data in public cloud," *Indian Journal of Science and Technology,* vol. 8, p. 1, 2015.

[69] A. IRUDAYASAMY and L. Arockiam, "SCALABLE MULTIDIMENSIONAL ANONYMIZATION ALGORITHM OVER BIG DATA USING MAP REDUCE ON PUBLIC CLOUD," *Journal of Theoretical & Applied Information Technology,* vol. 74, 2015.

[70] K. Pandilakshmi and G. R. Banu, "An Advanced Bottom up Generalization Approach for Big Data on Cloud," *Volume,* vol. 3, pp. 1054-1059, 2014.

[71] M. Balusamy and S. Muthusundari, "Data anonymization through generalization using map reduce on cloud," in *Computer Communication and Systems, 2014 International Conference on*, 2014, pp. 039-042.

[72] 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.

[73] B. B. Mehta and U. P. Rao, "Privacy preserving big data publishing: a scalable k-anonymization approach using MapReduce," *IET Software,* vol. 11, pp. 271-276, 2017.

[74] X. Zhang, L. T. Yang, C. Liu, and J. Chen, "A scalable two-phase top-down specialization approach for data anonymization using MapReduce on cloud," *IEEE Trans Parallel Distrib Syst,* vol. 25, 2014.