JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 20, No. 1, pp. 199-206, Jan. 2019

# 분산 환경에서의 웹기반 *k*-익명화 시스템

김 태 수 · 김 종 욱*
상명대학교 컴퓨터과학과

# Web-based *k*-Anonymization System in a Distributed Environment

**Tae-Su Kim · Jong Wook Kim***
Department of Computer Science, Sangmyung University, Seoul 110-743, Korea

**[요 약]**

오늘날 우리는 빅 데이터의 시대에 살고 있다. IoT, 모바일 기기의 발전 등으로 많은 분야에서 빅 데이터가 생성된다. 빅 데이터 시대에 맞게 빅 데이터를 관리하고 배포하는 기술의 중요성도 커지고 있다. 빅 데이터에는 수많은 정보를 포함하는데 정보 중에는 개인정보 또는 민감 정보를 포함하기 때문에 빅 데이터가 유출되면 심각한 개인정보침해를 초래한다. 개인정보를 익명화하는 대표적인 방법으로 *k*-익명화가 있다. 빅 데이터를 *k*-익명화를 통해 보안성을 높이고 만약 유출되어도 빅 데이터를 안전하게 보호할 수 있다. 본 논문에서 제시하는 *k*-익명화 시스템은 웹 환경에서 사용자가 자신의 데이터를 익명화 할 수 있는 환경을 제공한다. 사용자는 데이터의 구분자, 분류트리를 입력하여 데이터를 일반화한다. 일반화 데이터를 토대로 *k*-익명화 옵션 값을 설정하고 웹을 통해 서버로 전송한다. *k*-익명화 옵션이 서버로 전달되면 마스터 서버는 분산 환경을 통해 여러 워커 노드들에게 *k*-익명화 작업을 요청한다. *k*-익명화가 끝나면 결과물을 사용자가 다운받고 *k*-익명화된 데이터를 공유, 배포할 수 있도록 지원한다.

**[Abstract]**

Today we live in the era of Big Data. The ever increasing integration of IoT and mobile devices in many fields is resulting into generation of enormous volumes of data on daily basis. The importance of managing and distributing this big volumes of data is also increasingly growing. Big data contains a lot of information, some of which are personal and sensitive, and can result into violation of personal privacy incases of data leakage. Anonymizing big data is one of the several ways aimed at improving data security and privacy protection during data leakage scenarios. *k*-anonymization is one of the several techniques used for data anonymization. In this paper, we present a *k*-anonymization system which provides a suitable web environment for users to anonymize their data. The user generalizes the data by inputting a data delimiter and a taxonomy tree. Based on the generalization data, the *k*-anonymization option value is set and transmitted to the server via the web. When the *k*-anonymization option is passed to the server, the master server requests *k*-anonymization from multiple worker nodes through a distributed environment. After *k*-anonymization, users can download the output and share and distribute *k*-anonymized data.

색인어 : *k*-익명화, Spark, Hadoop, 분산 시스템, 데이터 보안
**Key word** : *K*-Anonymity, Spark, Hadoop, Distribued System, Data Privacy

**http://dx.doi.org/10.9728/dcs.2019.20.1.199**



**Received** 11 December 2018; **Revised** 25 December 2018
**Accepted** 20 January 2019

***Corresponding Author; Jong Wook Kim**

**Tel:** +82-2-781-7590
**E-mail:** jkim@smu.ac.kr

## Ⅰ. Introduction

Today, the amount of data generated is increasing on daily basis due to the IT related technological developments such as the emergence of IoT, the development of mobile devices, and the activation of SNS. This has resulted in the term "big data" and the value attached to this data is higher than in the past. As the data volume keeps growing, technological developments for managing and leveraging this data are becoming increasingly important.

Big data contains lots of information and some of which are person-specific. During data sharing or distribution, intentional or accidental data leakages by third parties or attackers can result into exposure of personal data and cause tremendous damage to personal privacy. Thus, data protection technologies are of significant importance during data sharing and distributions processes. Data anonymization technique has been suggested and used to provide the much desired data protection[1]-[4].

The most common method for data anonymization is *k*-anonymization[5]-[8], [13], [14]. To anonymize data, identifies data by dividing it into three attributes: an identifier, a quasi-identifier, and a sensitive attribute. An identifier identifies a particular individual with one piece of information, and a quasi-identifier identifies a particular individual by combining it with one or more pieces of information in the data and background knowledge or external data from a third party. Sensitive attributes are sensitive data of the user's information. For *k*-anonymization, the identifier is removed and the quasi-identifier is generalized to a certain category. The generalized data forms an equivalence class the category. If the number of records belonging to each equivalence class satisfies k or more, *k*-anonymization condition is satisfied and data is *k*-anonymized. In the event the *k*-anonymized data gets leaked, the number of internal records forming the equivalence class is k, so that a specific individual can not be identified. Thus, data distributors can securely distribute and share data.

Due to the continuos increase in the volume of big data, data processing time becomes an important metric to consider. To quicken up the data processing task and improve efficiency, distributed processing method is being used. Distributed processing saves big data processing time by distributing and processing big data in multiple nodes. Hadoop and Spark are representative distributed processing techniques[9], [11]. Hadoop distributes the big data to nodes consisting of n clusters and distributes them through hadoop mapReduce[10]. Spark is a general-purpose distributed platform that is similar to hadoop. Unlike hadoop, spark is much faster than hadoop. While hadoop processes data on disks, spark load the entire data set into memory and process it in memory.

In this paper, we develop a spark reliant, web-based *k*-anonymization system. The user sets the *k*-anonymization option value for his data on the web and passes the data and *k*-anonymization options to the server, which performs *k*-anonymization task using hadoop and spark and distributes the anonymized data to the users. The organization of the paper is as follows. In section II, we present the background of this study, section III describes the *k*-anonymization based on spark hadoop environment using the web. In section IV, we describe the *k*-anonymization system developed in this work through a case study. In section V, we present the conclusion.

## Ⅱ. Background Knowledge

In this section, we present the backgrounds on the *k*-anonymization technique, hadoop's hadoop distributed file system (HDFS) and yet another resource negotiator(YARN) functions, spark cluster structure and distributed processing.

### 2.1 *K*-Anonymity

A piece of data contains a lot of information. An attribute that identifies a particular individual in this data is called an 'identifier'. Examples of identifiers are; name, resident registration number, and so on. If a third party knows the identifier of the data, the personal information can be leaked. Therefore, in data sharing, it is necessary to delete identifiers before sharing the data. However, even if the identifier is removed from the data, there are still a significant number of attributes that can identify a particular individual in combination with external data, the attacker's background knowledge, and so on. An attribute that is not identifiable by a single piece of data but is likely to identify an individual through combination with other data is called a 'quasi-identifier'. Thus, in the data sharing, it is necessary to anonymize the quasi-identifier and as well as eliminate the identifier. The *k*-anonymization technique, which is one of the most common privacy model, requires the size of a equivalence class, which is a set of records having the same attribute value for a quasi-identifier, to be k or more, Making them indistinguishable [5], [7]. For example, Table 1 shows an original data table and an anonymized data table, and the quasi-identifiers correspond to 'Age' and 'Gender'. There are two equivalence classes in the anonymization table. In the anonymous table, each record is indistinguishable from other records in the equivalence class. For example, record 1 (can't) can not be distinguished from the other two records (RID = 4, 5). A generalization technique is used as a

**표 1.** 원본 테이블과 익명화 테이블
**Table 1.** Original table and an anonymized table

| (a) Original Table | | | |
|---|---|---|---|
| RID | Age | Gender | Disease |
| 1 | 32 | M(1) | Diabetes |
| 2 | 22 | M(1) | Anemia |
| 3 | 27 | F(0) | Pneumonia |
| 4 | 37 | M(1) | Anemia |
| 5 | 35 | F(0) | Diabetes |

| (b) Anonymized Table | | | |
|---|---|---|---|
| RID | Age | Gender | Disease |
| 1 | 30~39 | *(0~1) | Diabetes |
| 2 | 20~29 | *(0~1) | Anemia |
| 3 | 20~29 | *(0~1) | Pneumonia |
| 4 | 30~39 | *(0~1) | Anemia |
| 5 | 30~39 | *(0~1) | Diabetes |

typical method of transforming data in *k*-anonymization. The generalization method is a method of anonymizing data by replacing original data with generalized data according to generalization rules[5], [7]. The generalization rule is a rule applied to the generalization method and is based on a taxonomy tree of each attribute. The category tree is a hierarchical tree, and the higher the level, the higher the generalization level. For example, Figure 1 is a category tree for the attributes 'Age' and 'Gender'. As shown in Figure 1, the category tree for 'Age' has three levels of generalization, and the category tree for 'Gender' has two levels of generalization. As shown in the figure, the lower the generalization level, the closer it is to the original value, and the higher the generalization level, the wider the generalized value range. Therefore, lowering the level of generalization will increase the utilization of the data, but with a cost of increasing the risk of personal information leakage. On the other hand, increasing the level of generalization reduces the risk of personal information leakage at the cost of lowered data utilization. Thus, the key to the global generalization method lies in constructing appropriate generalization rules according to the type of the appropriate category tree and its form. Also, as shown in the category tree corresponding to 'Gender' in Figure 1, categorical data can be expressed as numerical data.

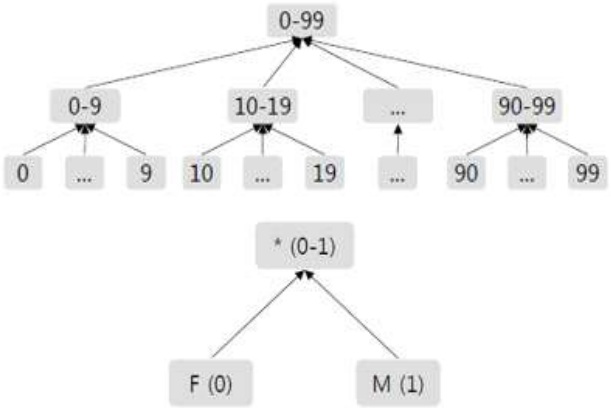


**그림 1.** Age와 Gender의 분류트리
**Fig. 1.** Example taxonomy trees of Age and Gender

## 2.2 Apache Hadoop

**1) Hadoop Distributed File System (HDFS)**

Hadoop is a software framework that can store and process big data. Hadoop uses HDFS, which is a file system that can distribute data across multiple servers[9]. HDFS is a block-structured file system that stores data in blocks. If a file has a size larger than that of a single disk, the file is divided into blocks, and the blocks are distributed to disk clusters connected to the HDFS.

HDFS is a master-worker structure, consisting of a name node and a data node. The name node manages the name space of the file system. It knows the location of all the blocks belonging to the file. The data node stores the data block and searches for and stores the block if requested by the name node. Also, it periodically transmits a list of data blocks stored in the data node to the name node. Figure 2 shows the structure of the name node and the data node.

**2) Yet Another Resource Negotiator (YARN)**

YARN is a Hadoop cluster resource management system. In the Hadoop architecture, YARN exists between HDFS and applications for big data. As shown in Figure 3, YARN consists of resource manager, node manager, application master, and container. The resource manager manages the entire cluster and coordinates the use of CPU, disk, memory, etc. among the applications. One node manager exists for each node, and

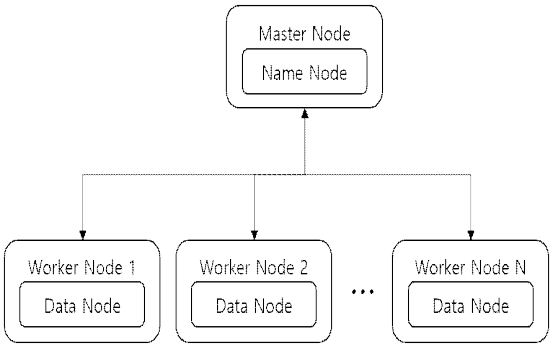


**그림 2.** HDFS 구조
**Fig. 2.** HDFS Architecture

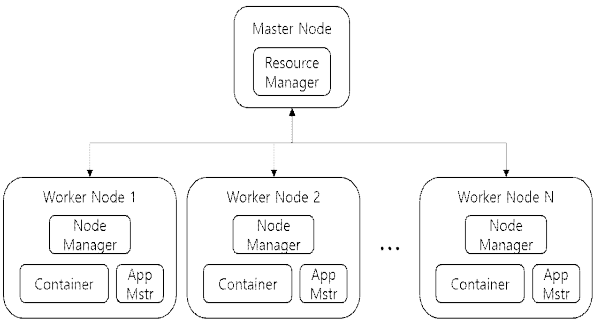


**그림 3.** YARN 구조
**Fig. 3.** YARN Architecture

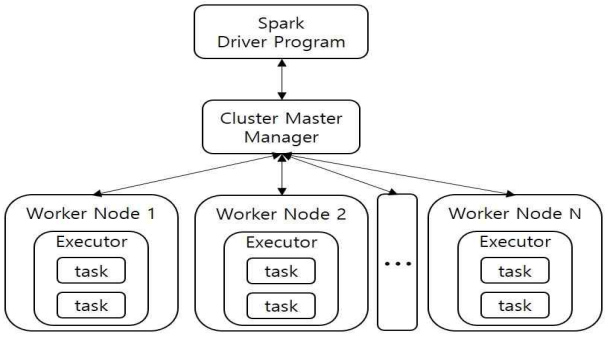


**그림 4.** Spark 구조
**Fig. 4.** Spark Architecture

monitors and manages resources of the worker node. Creates a container according to the resource manager's job requirements. Application masters are created one per task and use containers to monitor and execute tasks. They as well coordinate resource managers with resource requirements. Containers are defined by attributes such as resources. Every job is subdivided into several jobs, each job is executed in a container.

### 2.3 Apache Spark

Spark is a computing platform for in-memory-based clusters, unlike hadoop, which handles work on disks. Spark has many advantages over hadoop using its in-memory distributed processing, which is faster than hadoop's map & reduce, and supports multiple languages.

resilient distributed data set (RDD) is a collection of unchangeable data elements that are distributed. Spark distributes the data in the RDD to the clusters and executes each cluster RDD operation in parallel. RDD supports two types of operations: transformation and action. A transformation is an operation that creates a new RDD in an existing RDD. The action computes the result of the RDD operation and returns it to a driver program or to an external storage.

Every spark application consists of a driver program that performs parallel operations in the cluster. The driver program has the application's main function and defines the cluster's distributed data set. The driver program can access the spark through the SparkContext object and create an RDD. The spark operation consists of a stage which is an arbitrary directional acyclic graph (DAG) and is divided into a plurality of tasks to be distributed to each cluster. The delivered task is handled by an executor that executes on several nodes.

In distributed mode, spark uses a master-worker structure consisting of a driver program and multiple distributed work nodes. Server submit the application via 'spark-submit' and call the driver program's main function. The driver program asks the external service called cluster manager for resources to run the executor, and the cluster manager runs the executor and passes the work. In this paper, we use hadoop YARN as a cluster manager. When the job is finished, the results are passed to the driver program. When the main function of the driver program is terminated or the spark terminate command is called, the exciters are stopped and all resources are returned. Figure 4 shows the spark distribution structure.

## Ⅲ. System Structure

As shown in Figure 5, the *k*-anonymization system developed in this study consists of five steps.

(1) The user selects the data to be anonymized and inputs a delimiter to distinguish the contents of the data.
(2) The user selects the quasi-identifier attribute and the sensitive attribute excluding the identifier.
(3) An example taxonomy tree illustration is shown in Figure 8.
(4) Based on the input taxonomy tree, it outputs the equivalence class list and the number of each equivalence class of the

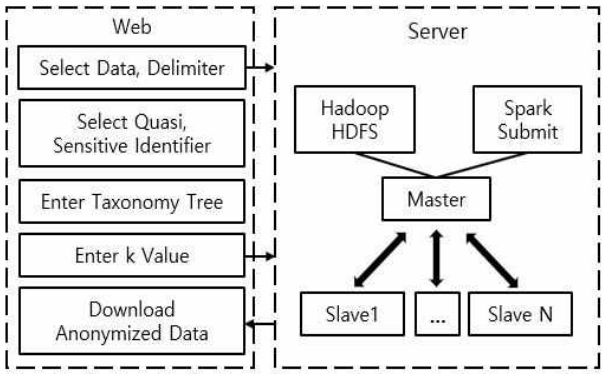


**그림 5.** k-익명화 시스템 수행 과정
**Fig. 5.** The progress of the *k*-anonymity System

sampled data generalized to the middle value of each taxonomy tree. The user sets the k value with reference to the sample data. The user proceeds to anonymization by sending the set k-anonymization option value to the server.

(5) When the k-anonymization is completed, the user is notified and the user then downloads the anonymized data.

A detailed description of all the steps is presented in section IV.

## IV. Case Study

For the case study presented in this section, we used the NPS dataset from the Health Insurance Review and Assessment (HIRA) service in Korea[12]. The National Patients Sample (NPS) dataset consists of electronic health records of 3% of the Korean people sampled in 2011. In this case study, the original table was generated by extracting six attributes ('Age', 'Sex', 'Location', 'Surgery', 'Length', and 'Disease') from the NPS dataset. We considered that the first five attributes corresponded to quasi-identifiers, and the attribute 'Disease' was a sensitive attribute. The table used in the case study consists of 3,000,000 records. We used a cluster with one master node and two slave nodes for the case study. Table 2 shows the specification of each node in the cluster and Spark options used in the case study

### 4.1 Data and Delimiter

As shown in Figure 6, the user selects the data to be anonymized and inputs a delimiter to divide each attribute of the data. The user-specified data is sent to the spark cluster. The data received from the user is divided into block units to be stored in the HDFS in the spark cluster and distributed to the nodes set as the data nodes of the HDFS.

### 4.2 Quasi-Identifier and Sensitive Attribute

**표 2.** 사례 연구에 사용 된 서버 사양과 Spark 옵션 값
**Table 2.** Server specification and Spark options used in the case study

| | | |
|---|---|---|
| Server Specs | HDD size | 4 TB |
| | Memory size | 64 GB |
| | CPU | 3.40GHz |
| | CPU cores | 8 |
| Spark Options | Executor-cores | 4 |
| | Executor-memory | 10 GB |
| | Driver-memory | 10 GB |
| | Java Heap Max | 55GB |

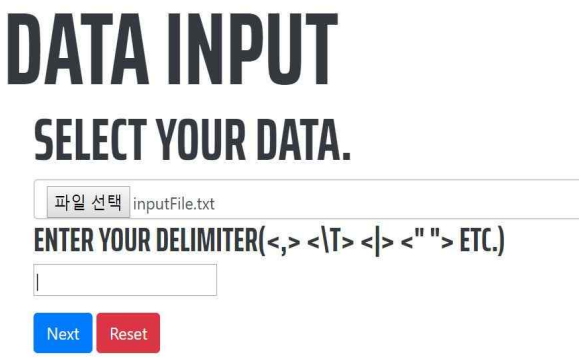


**그림 6.** 데이터 선택 화면
**Fig. 6.** User interface for selection of data

After completion of the input step, the user can read the description of the quasi-identifier and select the quasi-identifier and sensitive attribute excluding the identifier. A quasi-identifier corresponds to an attribute to be anonymized as described in Section 2.1. Figure 7 shows a screen snapshot of the user interface for selecting quasi-identifiers and sensitive attributes.

### 4.3 Taxonomy Tree

A screen snapshot for entering the taxonomy tree by the user for the data anonymization task is shown in Figure 8. The user can create a taxonomy tree by referring to the example of the taxonomy tree at the bottom left of the screen. A generalization lattice can be generated based on the taxonomy tree input by the user. The generalization lattice is made by the level of each property of the taxonomy tree that you enter. The number of cases from the initial level to the maximum value of each attribute makes the generalization lattice.


**그림 7.** 준 식별자와 민감 정보 선택 화면
**Fig. 7.** User interface for selecting quasi-Identifiers and sensitive attributes

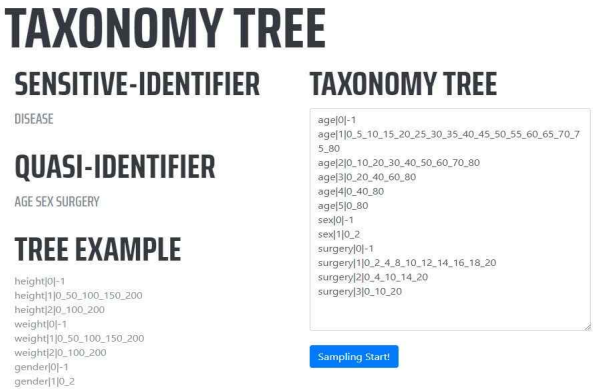


**그림 8.** 분류트리 생성 화면
**Fig. 8.** User Interface for Making Taxonomy Tree

### 4.4 *K*-Value

Generate a generalized lattice with intermediate values of each attribute level of the taxonomy tree input for sample data output and generalize the data. Outputs the equivalence class list and the number of equivalence classes of the generalized sample data to receive the k value of the equivalence class to the user. Enter the k value referring to the description of the equivalence class at the bottom of the screen and the result of the sample data of the user data. Figure 9 shows the input of k value based on equivalence class of sample data.

### 4.5 Download Anonymized Data

The web delivers the k value, quasi-identifier, sensitive attribute, data delimiter, and taxonomy tree that the user inputs to the server. The server creates and executes a spark application by adding the value sent to spark-submit. In the spark executor, the number of executors in the worker node is dynamically generated according to the state of the spark task queue. Because it iterates

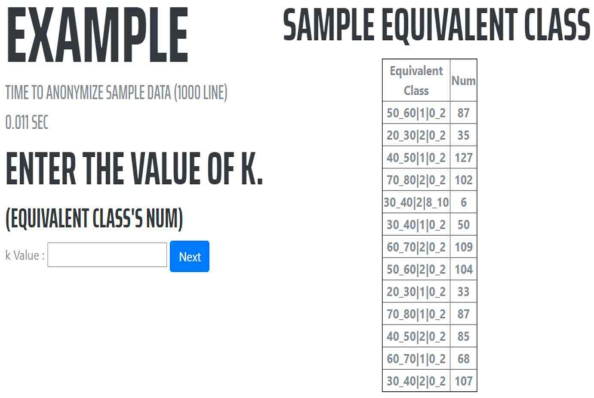


**그림 9.** k값 입력 화면
**Fig. 9.** User Interface for Entering k-Value

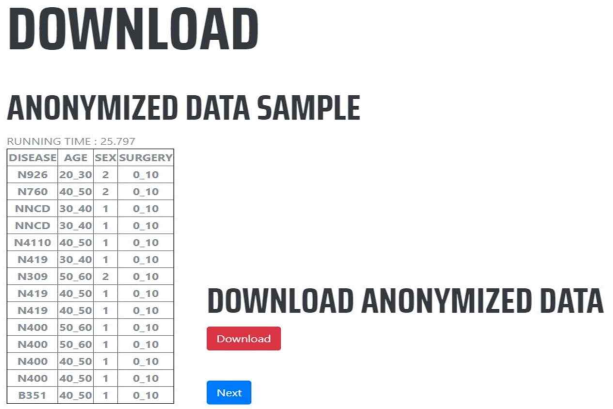


**그림 10.** 익명화 데이터 다운로드 화면
**Fig. 10.** User Interface for Downloading Anonymized Data

over the same data, it loads data into memory through spark cache instructions and performs in-memory work. It generalizes the blocks stored in each node using a generalized lattice. Then, the generalized data blocks are aggregated into one and the number of records of the equivalence class is counted. Then, compare of the number of all equivalence classes with the k value. If the number of equivalence classes is smaller than k, the k-anonymization criterion is not satisfied. Therefore, generalization is performed by changing the generalization lattice until the number of equivalence classes is greater than k. Even if the generalization is repeated, the data remains in the memory, so the data is read from the memory without reading the disk and generalization is performed with a new generalized lattice. If the number of records of all equivalence classes is greater than k, k-anonymized data is stored on the server and the user is notified of the completion of the anonymization operation. As shown in Figure 10, the user is presented with a sample of the anonymized data for confirmation and downloads it for distribution and sharing to others.

## Ⅴ. Conclusion and Future Work

In this study, we proposed a k-anonymization system reliant on Hadoop, Spark distributed processing and storage environment. The system presents users with interfaces for selecting the quasi-identifiers and sensitive attributes that the user desires and create data categories for each attribute in the taxonomy tree and output the generalized sample data. Based on the sample data presented, the desired k value can be set for the k-anonymization task to proceed. The anonymization process is very fast due to the distributed storage and processing environment provided by the Hadoop and Spark, and thus, an improvement on efficiency which

is vital in the big data era. Future research will focus on increasing the security by applying encryption during k-anonymization.

## Acknowledgements

This research was supported by a 2017 Research Grant from Sangmyung University.

## Reference

[1] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets", *In Proceedings of the 2008 IEEE Symposium on Security and Privacy Page*, Oakland, CA, USA, pp. 111-125, 2008.

[2] J. Kim, K.Jung, H. Lee, S. Kim, J.W. Kim and Y.D. Chung, "Models for Privacy-preserving Data Publishing : A Survey", *Journal of KIISE*, Vol. 44, No. 2, pp. 195-207, February 2017.

[3] B.C.M. Fung, K. Wang, R. Chen, and P.S. Yu, "Privacy-preserving data publishing: A survey of recent developments", *ACM Computing Surveys*, Vol. 42, No. 4, June 2010.

[4] N. Mohammed, B.C.M. Fung, P.C.K. Hung, and C.K. Lee, "Centralized and distributed anonymization for high-dimensional healthcare data", *ACM Transactions on Knowledge Discovery from Data*, Vol. 4, No. 4, October 2010.

[5] L. Sweeney, "k-anonymity: A model for protecting privacy", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 10, No. 05, pp. 557-570, May 2002.

[6] L. Sweeney, "Achieving k-Anonymity Privacy Protection using Generalization and Suppression", *International Journal of Uncertainty*, Vol. 10, No. 05, pp. 571-588, May 2002.

[7] K. LeFevre, D.J. DeWitt and R. Ramakrishnan, "Incognito: Efficient full domain k-anonymity", *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, Baltimore, Maryland, USA, pp. 49-60, 2005.

[8] J. Byun, A. Kamra, E. Bertino, N. Li, "Efficient k-Anonymization Using Clustering Technique", *DASFAA 2007: Advances in Databases: Concepts, Systems and Applications*, Bangkok, Thailand, pp. 188-200, 2007.

[9] Apache Hadoop 2.8.4 API docs[Internet]. Available: https://hadoop.apache.org/docs/r2.8.4/.

[10] Jens Dittrich, Jorge-Arnulfo Quiané-Ruiz, "Efficient big data processing in Hadoop MapReduce", *Proceedings of the VLDB Endowment*, Vol. 5, No. 12, pp. 2014-2015, August 2012.

[11] Apache Spark 2.3.0 API docs[Internet]. Available: https://spark.apache.org/docs/2.3.0/index.html.

[12] Health Insurance Review and Assessment Service in Korea[Internet]. Available: http://opendata.hira.or.kr.

[13] T. Kim, J.W. Kim, "Efficient K-Anonymization Implementation with Apache Spark", *Journal of The Korea Society of Computer and Information*, Vol. 23, No. 11, pp. 17-24, November 2018.

[14] J. Kim, K. Jung, H. Lee, S. Kim, J.W. Kim, and Y.D. Chung, "Models for Privacy-preserving Data Publishing : A Survey", *Journal of KIISE*, Vol. 44, No. 2, pp. 195-207, 2017.

**김태수**(Tae-Su Kim)

2014년 ~ 현 재: 상명대학교 (학사)

※관심분야 : 빅데이터(Big Data), 분산시스템(Distributed System)

**김종욱**(Jong Wook Kim)

2000년 : 고려대학교 (학사)
2002년 : KAIST (석사)
2009년 : Arizona State University (박사)

2009년~2010년: Technicolor
2010년~2013년: Teradata
2013년~현 재: 상명대학교 컴퓨터과학과 교수
※관심분야 : 분산 시스템(Distributed System), 데이터 보호(Data Privacy)