Journal of Harbin Engineering University
ISSN: 1006-7043
Vol 45 No. 6
June 2024

Development of Threat Information De-identification Technology Using Big Data Tools

Seung-Yeon Hwang[1], Jeong-Joon Kim[2,*]
[1]Dept. of Computer Engineering, Anyang University, Anyang-si, Gyeonggi-do, Republic of Korea
[2,*] Corresponding Author, Dept. of Software, Anyang University, Anyang-si, Gyeonggi-do, Republic of Korea

Abstract
In cybersecurity, many new technologies such as big data tools have developed, causing enormous damage to companies and public institutions not only through data generation but also major cyberattacks such as ransomware that abuse them. As threat information such as personal information is leaked by malicious hackers, cases requiring money such as voice phishing, and various spam emails and illegal telemarketing are used to constantly receive advertisement information that users do not want. The scale of mental and material damage continues to increase due to personal information leakage, and once leaked personal information is virtually impossible to recover, it is identified as important threat information. Therefore, in this paper, a de-identification technology that can safely store personal information among threat information was developed. De-identification technology provides strong security compared to general leaks because even if some or all the information is deleted or replaced with other characters, important parts are concealed. The de-identification technology operates and stores in MongoDB's sharding environment, and memory-based Spark is used because distributed parallel processing is required to de-identify large amounts of data. As the main data generated for testing this technology, personal information data in the form of integers, strings, and patterns can be generated as much as desired, and de-identification techniques such as suppression and masking can be applied through a convenient UI.
Keywords: Cybersecurity, De-identification, Personal information leakage, Distributed parallel processing

1. Introduction

Due to rapid development and rapid growth in today's information age, big data-based information systems are also rapidly developing, and now we can see a realistic future through the Internet of Things [1-2]. With the improvement of Internet-based technology through various systems in our living environment, anyone can easily access computer systems as they are in a living environment where the Internet is easily accessed, and networks are also developed [3-4]. In addition, companies and public institutions also require control and control of systems that develop day by day with the introduction of information systems. In addition to systematizing a system that is complexly set due to personal information or linkage of many systems, the system should be operated by setting it appropriately according to environment and use [5]. Attackers targeting companies or public institutions also attack in various patterns, and now the form of attacks is changing from individuals to groups through collectivization, and they need to know a lot of information knowledge along with the knowledge of professional information systems [6].

Recently, using big data in all industries in each country can provide customized marketing to customers [7]. Encrypting and storing personal information in a database reduces efficiency and availability, so to utilize big data, personal information must be de-identified so that third parties cannot recognize it [8]. Therefore, this paper directly implemented data suppression and masking methods among the main functions of de-identification and developed a de-identification technique that can generate sample data in the form of integers, strings, and patterns to test them. MongoDB, which can conveniently manage Json data among big data tools, is selected as the storage of the developed technology and is stored in Sharding, a distributed environment, to prepare for failure recovery of generated and de-identified data. In addition, it operates in a Spark environment for fast parallel processing, and the programming language actively utilizes PySpark, a Spark-based Python. PySpark was used as the processing part

that requires operation in the Spark distributed environment, and it was developed by fusion like a general Python language. In addition, through the UI, users can easily generate data in the form of integers, strings, and patterns, apply suppression and masking de-identification techniques, and there is also a data viewer menu to compare data before and after de-identification.

Starting with Section 1 Introduction, Section 2 introduces the concept of de-identification developed in this paper and related technologies Spark and MongoDB, while Section 3 introduces the architecture, environment, and main functions of the de-identification technology developed. Finally, it concludes by summarizing this paper in the conclusion.

2. Related Works

2.1 De-identification

De-identification is a process used to prevent someone's identity from being disclosed. For example, data generated during human experiments can be de-identified to protect the privacy of study participants [9]. When applied to metadata or general data on identification, this process is also called data anonymization. Common strategies include deleting or masking personal identifiers such as lives and omitting or generalizing quasi-identifiers such as birth dates. The process of using de-identified data in reverse to identify individuals is called data re-identification [10]. De-identification is adopted as one of the main approaches to data privacy protection. It is commonly used in the fields of communication, multimedia, biometrics, big data, cloud computing, data mining, and social network audio-video monitoring. The main techniques for de-identification can be summarized as shown in Tab 1.

Tab. 1. Major techniques for de-identification

Technology Name	**Content**	**Method**
Pseudonymization	Replace with other values that cannot be directly identified for personal identifiable data.	Heuristic Pseudonymization Encryption Swapping
Aggregation	Statistics (all or part) are applied to personal information so that a specific individual cannot be judged.	Aggregation Micro Aggregation Rounding Rearrangement
Data Reduction	Delete specific data values that can identify personal information.	Reducing Variables Reducing Partial Variables Reducing Records Trivial Anonymization
Data Suppression	To prevent tracking and identification of unique information by converting (categorizing) single identification information into a representative value of a corresponding group or converting (scoping) it into an interval value.	Data Suppression Random Rounding Data Range Controlled Rounding
Data Masking	The personal identification information is entirely or partially converted into a substitution	Adding Random Noise Blank and impute

	value (space, '*', noise, etc.).	

Table 1 summarizes the de-identification technologies and applies the method corresponding to each technical name to the de-identification technology developed in this paper. The four detailed techniques of Data Suppression: data suppression, random rounding, data range, and control rounding were all implemented, and adding random noise, blank and impute were all implemented in masking. Therefore, the de-identification technology of this paper provides all data suppression and masking functions.

2.2 Spark

Spark emerged to overcome the limitations of the MapReduce cluster computing paradigm. MapReduce reads the data from the disk, bundles the data scattered through Map between the relevant data in the form of a key-value, removes the duplicate data through Reduce, processes it to the desired data, and stores it back on the disk. However, these file-based disk I/O performance was poor, and Spark emerged to improve processing performance through In-memory operations [11].

Apache Spark is an open-source, universal purpose distributed cluster computing framework that helps you program clusters with fault tolerance & data parallelism. Apache Spark offers three APIs: RDD, Data Frame, and Data Set, which enable in-memory operations based on these data, boosting performance by about 100 times compared to disk-based HADOOP. There are three typical ways to build Spark into a cluster environment: Spark Standalone Cluster, Hadoop Yarn Cluster, and Apache Mesos Cluster [12].

Spark Standalone Cluster is a mode that uses Spark's own Cluster Manager and can be configured most easily. In addition, it consists of one master and several workers, each using as much memory and the number of CPU cores according to settings. The master plays the role of a cluster manager and allocates necessary server resources at the request of a client, and manages the execution of a worker's Worker runs Executor and Task to perform actual processing and storage of data, and for Spark Standalone clusters, Executor operates one per worker node.

Hadoop Yarn Cluster is a way that YARN acts as a cluster manager to run and manage applications, and Yarn is automatically installed together when Hadoop is installed. If you run the spark on YARN, it can be useful when storing or importing data because the spark operates on the HDFS where the desired data is stored.

The Apache Mesos Cluster consists of a master who manages Slave daemons running on each node and a framework for running tasks on these Slaves. The framework running on Mesos consists of two components, the first one being divided into a scheduler registering the resources provided to the master and a process starting on the Slave node to execute the framework's tasks. While Master determines the number of resources to provide to each framework, the scheduler in the framework selects the resources to be used among the resources provided, and once the framework accepts the resources provided, it communicates to Mesos about the work it wants to start.

Among the above three methods, Spark Standalone Cluster is used as the cluster used to build a real-time parallel processing environment. This is because there is no need to install Hadoop separately to build this system environment, and the size of the data used itself is not large, which may cause more load if managed through resources. The operation process of Spark Standalone Cluster proceeds as SparkContext connects to Cluster Manager, receives resources allocated, transmits libraries to be used for allocated resources, and passes tasks to be executed.

2.3 MongoDB

MongoDB is a NoSQL database and is a document-oriented database that stores JSON-type data. With databases designed for web applications and the Internet, data models and persistence strategies were created with high read/write efficiency and ease of expansion through Failover in mind.

Many developers use MongoDB because data models are more convenient than typical SQL-based databases because data values rather than scalability are stored in documents rather than Row values. MongoDB's document format is based on JSON, well known as a schema that stores any data structure. JSON stands for JavaScript Object

Notation, and MongoDB uses documents in the form of JSON, which is frequently used when communicating with web servers, unlike how RDBMS store data in table format. JSON, along with XML, is the basic form of data exchange used on the modern web, and JSON supports all basic data types such as numbers, strings, and boolean values, as well as arrangements and hash. The structure of JSON consists of a key and a value for the key, and there is no limit to overlapping. The document-based data model may represent data of a rich and hierarchical structure [13].

MongoDB does not support SQL, so it does not have a join concept, and because the schema is flexible, it has the convenient advantage of not having to have complex join operations between multiple tables required by RDBMS. The data unit stored in MongoDB is a document, which can be said to be a record of row unit in RDBMS. Therefore, because MongoDB's document properties are not standardized like SQL and are variable, all document types are suitable for storing and processing unstructured data. When programming using object-oriented languages, the complexity of the object mapper disappears because the objects defined in the programming language are stored as they are [14].

The biggest advantage of MongoDB is that it basically provides replication and sharding. To prepare for failure and preserve data, sharding can be used to distribute information and speed up I/O. Since schema changes are free compared to conventional RDBMSs, it is useful to introduce MongoDB for models that are difficult to predict or are not finally determined. In addition, MongoDB is a good choice when a distributed computer environment is needed. The schemaless means that compared to RDBMS, it is relatively free from schema restrictions, not schema-free environments [15].

3. Methods

3.1 Architecture and Environment

The main architectural environments of Spark and MongoDB of the de-identification tool implemented in this paper are shown in Figures 1 and 2.

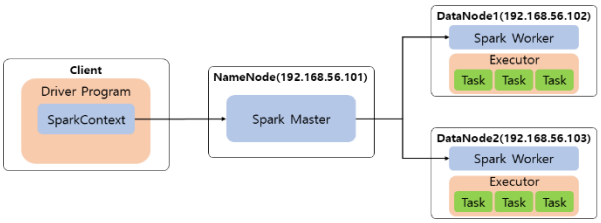


Fig. 1. Spark architecture to apply de-identification technology

Figure 1 is Spark's architecture applied with de-identification technology, which uses a total of three nodes (NameNode, DataNode1, and DataNode2) to process data in parallel in a distributed environment by issuing commands to the Spark Master of NameNode.

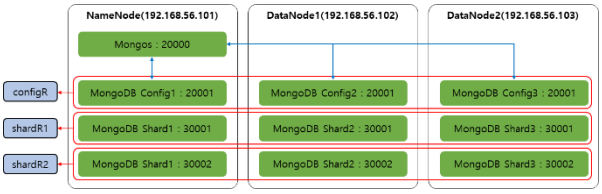


Fig. 2. MongoDB architecture to apply de-identification technology

Figure 2 is a structure designed to distribute, store, and parallelize databases in software to distribute and store large amounts of data. The MongoDB version used to build the environment is 4.2.8, and three nodes (NameNode-192.168.56.101, DataNode1-192.168.56.102, DataNode3-192.168.56.103) are used. And use the 20001 port number when running the Config daemon on each node, the 30001 port number when running the Shard1 daemon, and the 30002 port number when running the Shard2 daemon. In addition, 20000 port number is used to run Mongos daemon, which acts as a router for Sharding Cluster in NameNode.

The Config Server of each node serves to store the metadata required for distributed storage of each data, and before MongoDB 3.4 version, Config Server was configured alone, but since it must be configured as ReplicaSet from 3.4 version or higher, ReplicaSet is designated in the name of ConfigR. The Shard Server, where actual data is physically stored, configures it on each node and designates ReplicaSet under the names of shardR1 and shardR2. When Mongos Server configures the Sharding Cluster as a server serving as routing, the client accesses Mongos Server and enters a command, which sends a query to the Shard connected from Mongos Server to collect and notify the client again.

3.2 De-identification Technology

The automatic data generation function in the form of integers, which is a function that can automatically generate test data for verification in the form of integers to verify the de-identification function, consists of parameter setting, generation engine, and data inquiry.

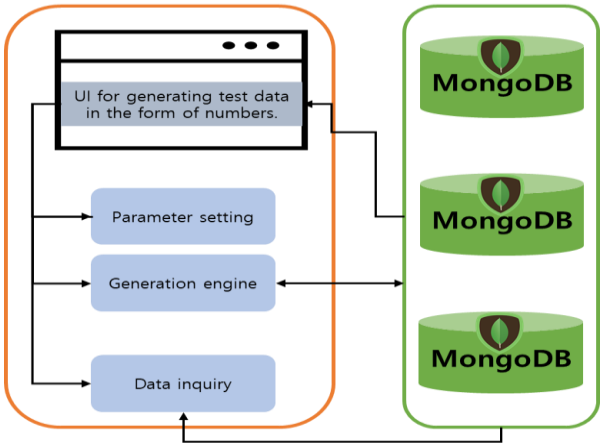


Fig. 3. Scenario for generating test data for verification in the form of integers

The term parameter setting means a function of setting a parameter for generating test data for verification in the form of an integer.

- Age: Enter the minimum and maximum values, which are the ranges of the age to be generated.
- Height: A minimum value and a maximum value, which are ranges of keys to be generated, are input.
- Weight: Enter the minimum and maximum values, which are the ranges of the weight to be generated.
- Generations: The integer of data fields to be generated is input.

The generation engine is a function for generating test data for verification in the form of integers, and refers to a function of generating age, height, and weight by receiving the minimum and maximum values input in parameter setting. In addition, data inquiry refers to a function of inquiring the generated numerical test data. Figure 4 shows the generation of test data as a UI screen for generating test data for verification in the form of integers.

As shown in Figure 4, the menu that generates test data in the form of integers can be selected by selecting integer, the first menu of the de-identification and verification tool program. When the data to be generated is age (20 to 30), height (150 to 190), and weight (40 to 80) and the minimum and maximum values of each data are input, the data is randomly generated within the input range of the minimum and maximum values. Figure 5 shows the data generated through data verification after creation.

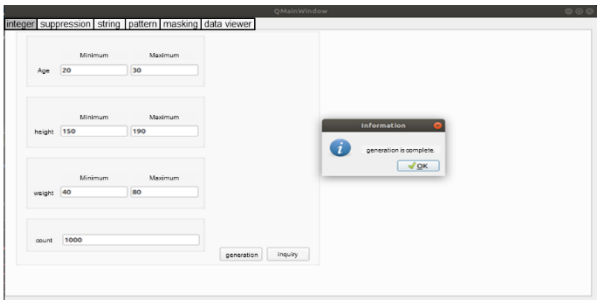
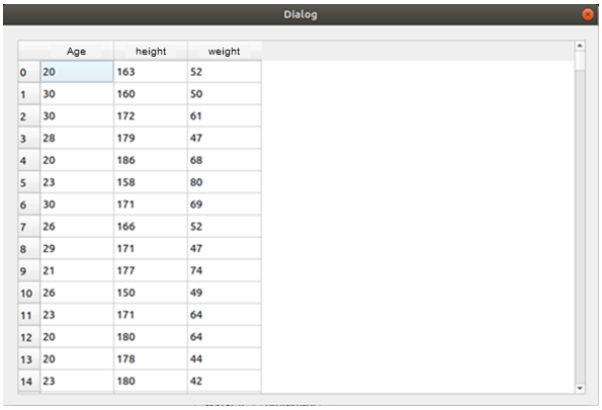


Fig. 4. Example of generating integer type test data



	Age	height	weight
0	20	163	52
1	30	160	50
2	30	172	61
3	28	179	47
4	20	186	68
5	23	158	80
6	30	171	69
7	26	166	52
8	29	171	47
9	21	177	74
10	26	150	49
11	23	171	64
12	20	180	64
13	20	178	44
14	23	180	42

Fig. 5. Checking the generation of data in the form of integers

The automatic data generation function in the form of a string, which is a function that can automatically generate test data in the form of a string to verify the de-identification function, is shown in Figure 6 as parameter setting, generation engine, and data inquiry.

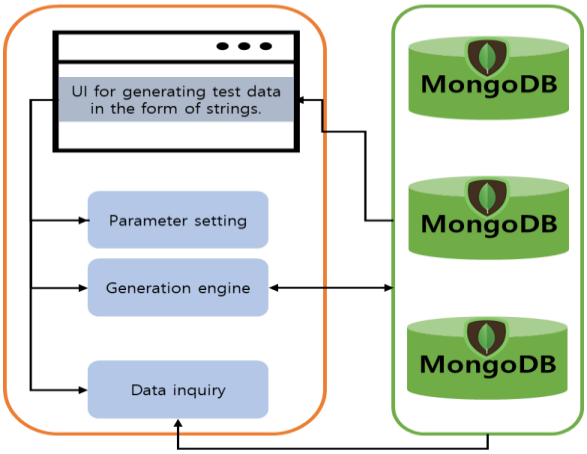


Fig. 6. Scenario for generating test data for verification in the form of string

Parameter setting refers to a function of setting a parameter for generating test data for verification in the form of a character string.

- Name: Define the family name and name data structure, and enter the number of name digits to be generated.
- Address: You can build and select an area to create address data in the MongoDB Sharding Cluster, and the areas you can select are randomly composed of Seoul, Incheon, Daejeon, Daegu, Busan, and so on.
- User ID: Enter the number of digits of the ID to be generated. ID is a combination of strings and numbers.
- Email address: defines the country and domain data structure, and selects the number of ID digits to be created, corresponding countries (kr, jp, us, cn, fr, random), and fields (edu, ac, go, co, or random).
- Generations: The number of data fields to be generated is input.

Generation engine is a function for generating test data for verification in the form of a string, and refers to a function of generating a name, address, user ID, and email address by receiving a value input or selected from parameter setting. Data inquiry refers to a function of inquiring test data in the form of a generated string. Figure 7 shows the generation of test data as a UI screen for generating test data for verification in the form of a string.

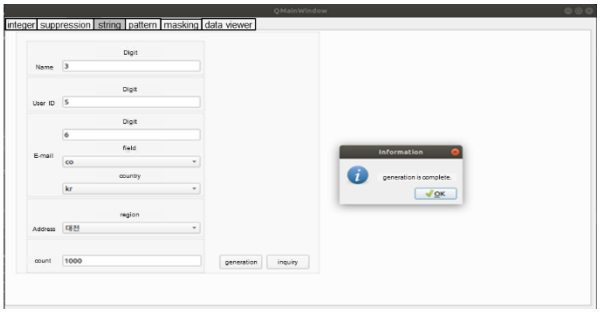


Fig. 7. Example of generating string type test data

As shown in Figure 7, the menu that generates test data in the form of characters can be selected by selecting string, the third menu of the de-identification and verification tool program. The data to be generated is entered as many numbers as the number of digits to be generated by name, ID, e-mail, and address, and in the case of e-mail, field combo boxes can select edu, ac, go, co, or random, kr, jp, us, cn, fr, random, and address combo boxes can select Seoul, Incheon, Daegu, Busan. An example of generating test data in the form of a string by selecting 3 digits of the name to be generated, 5 digits of the ID, 6 digits of the email ID, co in the field, kr in the country, and Daejeon for the address. Figure 8 shows the data generated through data verification after creation.

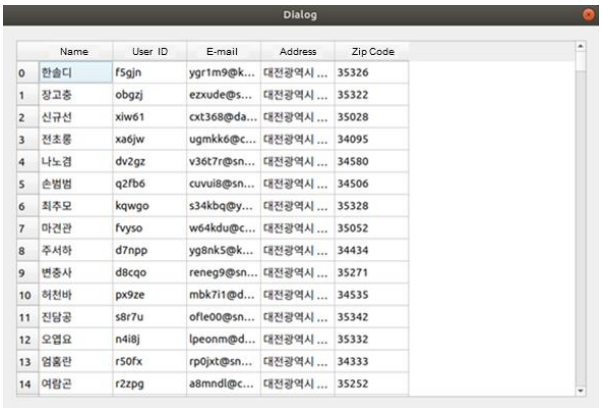


Fig. 8. Checking the generation of data in the form of strings

In Figure 8, if you look at the list created with three digits in the Name column, there is a value called "한솔디" at first. In Korea, the name is not an alphabet, so "한" means one digit, "솔" means one digit, and "디" means one digits.

The automatic data generation function in the form of a pattern, which is a function that can automatically generate test data in the form of a pattern to verify the de-identification function, consists of parameter setting, generation engine, and data inquiry.

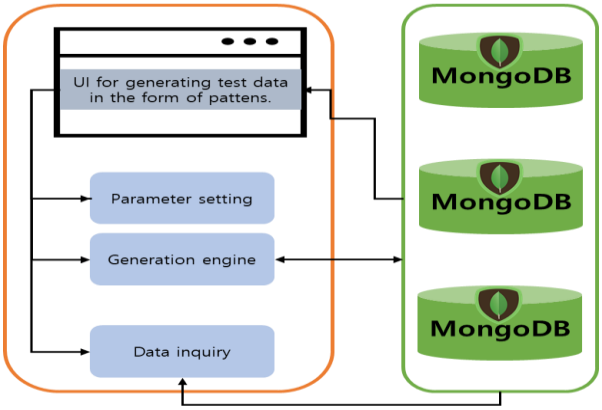


Fig. 9. Scenario for generating test data for verification in the form of pattens

The term parameter setting means a function of setting a parameter for generating test data for verification in a pattern form.

- Name: Define the family name and name data structure, and enter the number of name digits to be generated.
- Age: Enter the minimum and maximum values, which are the ranges of the age to be generated.

- Date of birth: The minimum and maximum values for the date (year, month, day) to be generated through the calendar button are selected in the UI.
- Resident registration number: In the UI, the minimum and maximum values for the date (year, month, day) to be created through the calendar button are selected, and the back digits vary according to gender, so male, female, and random values are selected.
- Phone number: You can select an area, and the area can be selected based on the area numbers of Seoul, Incheon, Daejeon, Daegu, and Busan, and for mobile phones, you can use a unique 010 number. In the case of random, a value is randomly selected.
- Account number: Enter the pattern of the account number to be created directly. In order to distinguish an intermediate number of account numbers, the number of account numbers is classified using a hyphen symbol ("-").
- Zip code: You can select an area, and you can choose between Seoul, Incheon, Daejeon, Daegu, and Busan, and random values will be specified if you select Random.

Generation Engine refers to the function of generating name, age, date of birth, resident registration number, phone number, account number, and zip code by receiving values input or selected in parameter setting as a function for generating test data in the form of a pattern. "Data inquiry" refers to a function of inquiring test data in the form of a generated pattern. Figure 10 shows the generation of test data as a UI screen for generating test data for verification in the form of a pattern.

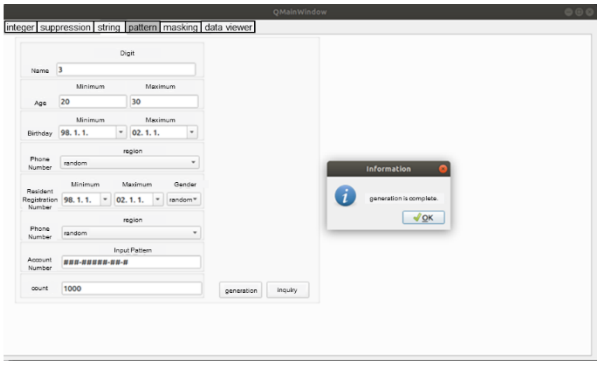


Fig. 10. Scenario for generating test data for verification in the form of pattens

As shown in Figure 10, the menu that generates test data in the form of a pattern can be selected as "Pattern", the fourth menu of the de-identification and verification tool program. The name is digit, the age and date of birth are minimum and maximum, and the phone number is a combo box. The resident number is based on the minimum and maximum values, and the back seat is distinguished by male/female/random, and the zip code is selected by combo box. Account numbers can be created by entering a certain pattern, and by pressing the right edge button of the input box on the date of birth, you can select them in the form of a calendar, and the same resident number is applied. You can choose Seoul, Incheon, Daejeon, Daegu, Busan, mobile phones, and random for the area of the phone number, while you can choose Seoul, Incheon, Daejeon, Daegu, Busan, and random for the zip code. The number of digits of the name to be created is 3, 20 to 30 of the age, and the date of birth is from 1998.01 to 2002.01.01. Phone numbers are random, and resident numbers are designated from 1998.01 to 2002.01.01. The postal code was random, and 1,000 account numbers were created by designating the pattern format of ###-#####-##-#. Figure 11 shows the data generated through data confirmation after creation.

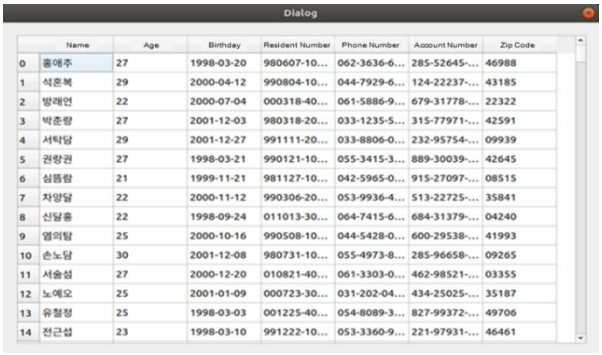


Fig. 11. Checking the generation of data in the form of patterns

In Figure 11, if you look at the list created with three digits in the Name column, there is a value called "홍애추" at first. In Korea, the name is not an alphabet, so "홍" means one digit, "애" means one digit, and "추" means one digits.

Data suppression functions, which are key processing functions for de-identifying sensitive data (age, height, weight, zip code, date of birth, resident registration number, address, e-mail address, etc.) from original data based on data suppression, data range, random rounding, and controlled rounding.

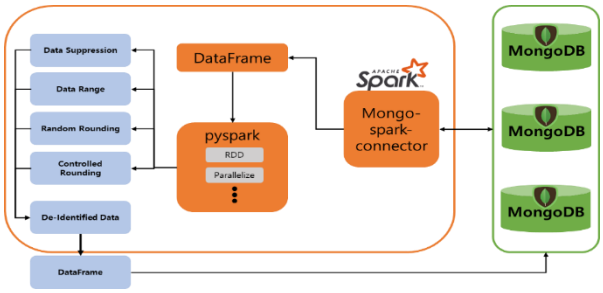


Fig. 12. Data suppression and de-identification functional scenario

Data suppression refers to a de-identification function that converts data into mean or category values to hide clear values. Data range refers to a de-identification function of setting numerical data for personal identification information to an arbitrary number and range. Random rounding refers to a de-identification function in which numerical data on personal identification information is raised based on an arbitrary number of digits. Controlled rounding refers to a de-identification function in which rows and columns match to solve the disadvantage that the sum of rows and columns does not match in the random rounding method. As shown in Figure 12, data stored in the distributed database MongoDB Sharding Cluster is read in the form of a DataFrame using the Spark-based Mongo-Spark-Connector module. The read data uses functions and methods such as Pyspark-based RDD and Parallelize to perform data suppression, data range, random rounding, and controlled rounding suppression-based de-identification functions, and the de-identified data is stored in the form of a data frame and distributed and stored again in MongoDB. Figure 13 shows an example of performing a data suppression-based de-identification function.

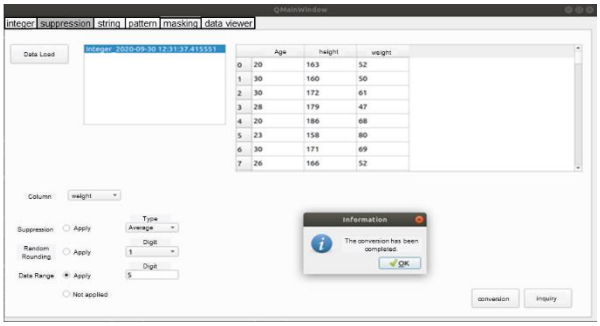
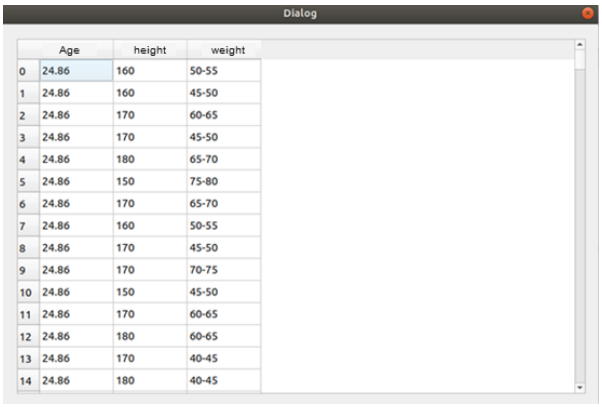


Fig. 13. Example of integer data suppression-based de-identification processing

As shown in Figure 13, the menu that performs the data suppression-based de-identification function can select suppression, the second menu of the de-identification and verification tool program. When the data load menu is selected, a list of test data collections in the form of numbers is retrieved. You can select the data you want to convert among the imported collections, suppression, random rounding, and data range for each column, and enter the places and units you want to suppression. Figure 13 brings up integer test data, ages are identified in five units of suppression average, height is one digit of random rounding, and weight is data range apply, and Figure 14 shows what was confirmed after de-identification.



	Age	height	weight
0	24.86	160	50-55
1	24.86	160	45-50
2	24.86	170	60-65
3	24.86	170	45-50
4	24.86	180	65-70
5	24.86	150	75-80
6	24.86	170	65-70
7	24.86	160	50-55
8	24.86	170	45-50
9	24.86	170	70-75
10	24.86	150	45-50
11	24.86	170	60-65
12	24.86	180	60-65
13	24.86	170	40-45
14	24.86	180	40-45

Fig. 14. Confirmation after de-identification processing based on integer data suppression

Data masking, a key processing function for de-identifying sensitive data (name, address, user ID, email address, age, height, weight, phone number, zip code, date of birth, resident registration number, account number, etc.) based on data masking, consists of adding random noise and blank and impute.

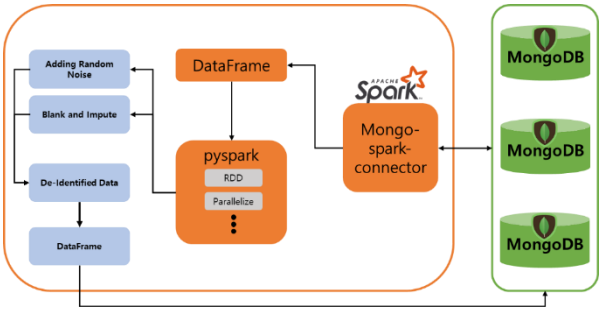


Fig. 15. Data masking and de-identification functional scenario

Adding random noise is a new anonymization method for sensitive personal identification items such as income, which means a de-identification function that prevents exposure of identification information by adding or multiplying random noise such as random numbers. Blanks and impute refer to de-identification functions that change some or

all of a specific item into blanks or alternative characters (full symbols such as '*', '#', etc.). As shown in Figure 15, data stored in the distributed database MongoDB Sharding Cluster is read in the form of a DataFrame using the Spark-based Mongo-Spark-Connector module. The read data uses functions and methods such as Pyspark-based RDD and Parallelize to perform masking-based de-identification functions such as adding random noise and black and impute, and the de-identified data is stored in the form of a data frame and distributed and stored again in MongoDB. Figure 16 shows an example of performing a data masking-based de-identification function.

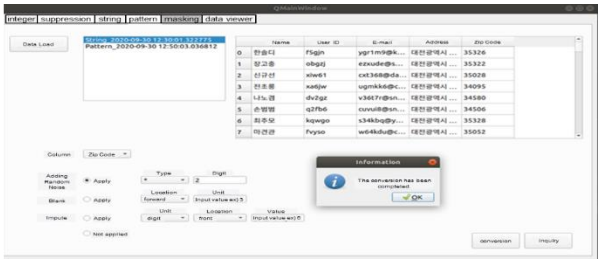


Fig. 16. Example of string data masking-based de-identification processing

As shown in Figure 16, the menu that performs the data masking-based de-identification function can be selected as masking, the fifth menu of the de-identification and verification tool program. When the data load menu is selected, a list of test data collections in the form of a string and a pattern is retrieved. Among the imported collections, data to be converted can be selected, adding random noise, space and impute can be selected for each column, input value calculation for integer form columns, and blank and impute can specify or input the location and unit to be masked for string and pattern forms. The method of masking and de-identifying string type test data is as follows. The name is input in units of 2 with the masking position of the blank function forward. The id has a masking unit of an impute function as a digit, and the masking unit inputs a value of 2. The email designates the masking unit of the impute function as an id, and the masking location is designated in the forward. The address has a masking unit of an impute function as a digit, and the masking position is designated forward to input a value of 5. The zip code selects a multiplication operation of the adding random noise function and inputs a value of 2. Figure 16 shows an example of string type test data being retrieved and de-identified according to the above input method, and Figure 17 shows the appearance confirmed after de-identification processing.

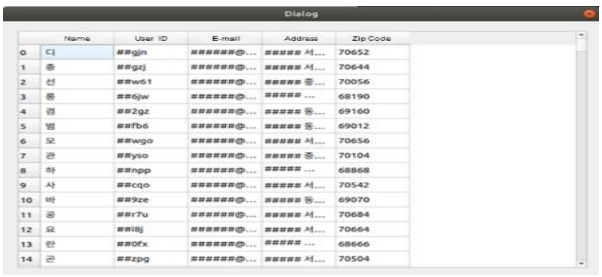


Fig. 17. Confirmation after de-identification processing based on string data masking

Figure 17 shows that the test data in the form of a string is de-identified through masking. Depending on the method previously processed in the string represented by "한솔디" in the name column, the "한솔" string is erased because it is a two-digit forward of the blank function, leaving only "디".

The masking de-identification processing method of pattern type test data is shown in Figure 18. The name is input in units of 2, leaving the masking position of the blank function behind. Age selects the multiplication operation of the adding random noise function and inputs 2 as the value. On the date of birth, the masking position of the blank function is moved forward, and 4 is input in units. The resident registration number is not applied without performing anything. The phone number puts the masking position of the blank function forward and inputs 4 in units. The account number has a masking unit of an impute function as a digit, and the masking location is designated forward and a value of 3 is input. The zip code selects a multiplication operation of the adding random noise function and inputs 2 as a value. Figure 19 shows how it was confirmed after de-identification.

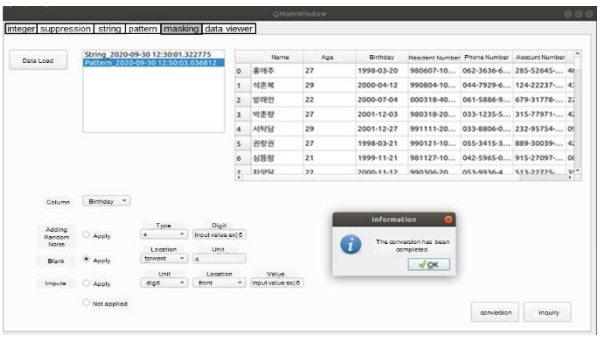


Fig. 18. Example of pattern data masking-based de-identification processing

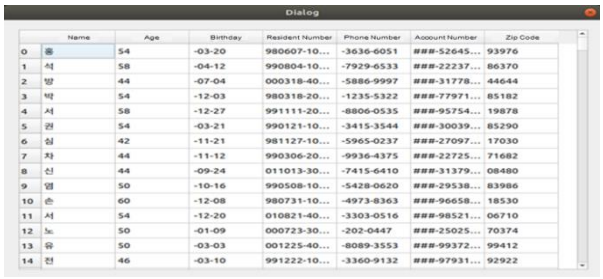


Fig. 19. Confirmation after de-identification processing based on pattern data masking

Before performing the de-identification function, the de-identification function verification UI, which can perform the de-identification function and compare the generated de-identification data with the original data, consists of Figure 20.

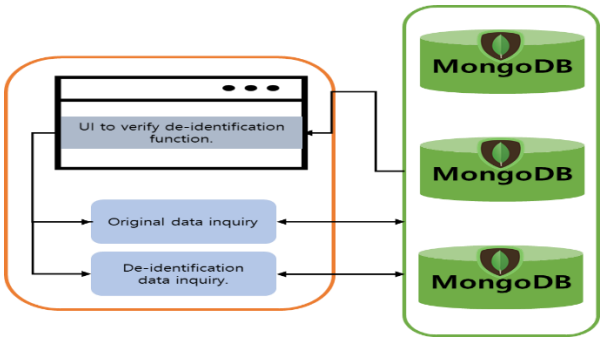


Fig. 20. Scenario for verifying de-identification

Original data inquiry shows a list of test data for verification in the form of generated strings, numbers, and patterns, and when selected, the original data can be searched by outputting it from the UI. The de-identification data inquiry shows a list of generated de-identification data, and when selected, it can be output from the UI to inquire about de-identification data. Figure 21 shows the appearance of verifying the de-identification function.

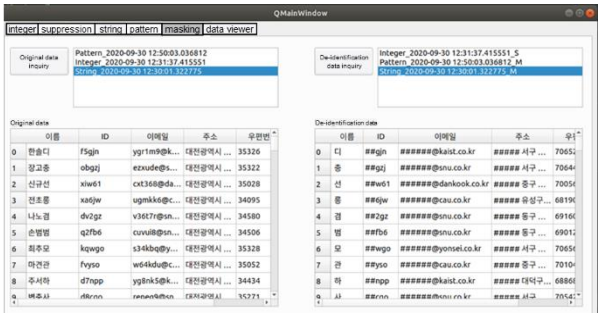


Fig. 21. Verification of de-identification function

As shown in Figure 21, the menu that performs the de-identification function verification function can select data viewer, the last menu of the de-identification and verification tool program. By original data inquiry from the data viewer, test data in the form of integers, strings, and patterns generated before processing de-identification can be retrieved. In addition, result data may be retrieved after performing suppression and masking functions through de-identification data inquiry. It can be confirmed that the data is the same through the time value at the end of the imported data and the de-identified data.

4. Conclusion

Various technologies and open source-based systems are being developed to store and manage big data at home and abroad, but existing technologies and tools are inefficient because they do not provide optimized storage and management methods for cyber threat information learning big data. It is necessary to establish an environment capable of distributed storage, parallel processing, and management considering the characteristics of cyber threat information learning big data, and to develop functions to analyze and visualize the results of collected learning big data. Since personal information can be exposed to cybersecurity threats and may have sensitive data, it is necessary to develop de-identification technologies to safely protect various learning data disclosed in the cyber strategy field.

Therefore, in this paper, we developed a technology that can de-identify personal information among threat information that is increasing day by day. Among many de-identification technologies, data suppression, data range, random rounding, controlled rounding methods were implemented in data suppression, and adding random noise, blank and impute methods were implemented in masking. In addition to the de-identification method, the function of automatically generating data in the form of integers, strings, and patterns was additionally implemented because test data to be applied were required. In addition, in order to confirm the normally converted matters after de-identification, a data viewer was also implemented to compare and view automatically generated data with de-identified data. Data generated and converted through the above technology is stored in sharding, a cluster distributed environment of MongoDB, in consideration of reliability and efficiency during failure recovery. The de-identification technology behavior used Python language, and Spark's

PySpark language was additionally fused to quickly distribute and parallelize data in memory. In addition to the de-identification technology currently implemented, we plan to implement additional techniques such as pseudonymization, aggregation, and data reduction, as well as the ability to generate more diverse forms of data divided into numbers, strings, and patterns according to user requirements.

Acknowlegement

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1F1A1062953).

References

[1] Lazer, D., & Radford, J. (2017). Data ex machina: introduction to big data. Annual Review of Sociology, 43, 19-39. https://doi.org/10.1146/annurev-soc-060116-053457

[2] Bendre, M. R., & Thool, V. R. (2016). Analytics, challenges and applications in big data environment: a survey. Journal of Management Analytics, 3(3), 206-239. https://doi.org/10.1080/23270012.2016.1186578

[3] Kim, Y. G., Kim, S. H., Jo, M. H., & Kim, W. J. (2014). The Bigdata Processing Environment Building for the Learning System. The Journal of the Korea institute of electronic communication sciences, 9(7), 791-797. https://doi.org/10.13067/JKIECS.2014.9.7.791

[4] Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., & Shahabi, C. (2014). Big data and its technical challenges. Communications of the ACM, 57(7), 86-94. https://doi.org/10.1145/2611567

[5] Alter, S. (2013). Work system theory: overview of core concepts, extensions, and challenges for the future. Journal of the Association for Information Systems, 72. https://doi.org/10.17705/1jais.00323

[6] Ben-Asher, N., & Gonzalez, C. (2015). Effects of cyber security knowledge on attack detection. Computers in Human Behavior, 48, 51-61. https://doi.org/10.1016/j.chb.2015.01.039

[7] Erevelles, S., Fukawa, N., & Swayne, L. (2016). Big Data consumer analytics and the transformation of marketing. Journal of business research, 69(2), 897-904. https://doi.org/10.1016/j.jbusres.2015.07.001

[8] Prasser, F., Kohlmayer, F., & Kuhn, K. A. (2016). The importance of context: Risk-based de-identification of biomedical data. Methods of information in medicine, 55(04), 347-355. https://doi.org/10.3414/ME16-01-0012

[9] Sweeney, L. (2000). Simple demographics often identify people uniquely. Health (San Francisco), 671, 1-34.

[10] Ribaric, S., Ariyaeeinia, A., & Pavesic, N. (2016). De-identification for privacy protection in multimedia content: A survey. Signal Processing: Image Communication, 47, 131-151. https://doi.org/10.1016/j.image.2016.05.020

[11] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... & Talwalkar, A. (2016). Mllib: Machine learning in apache spark. The Journal of Machine Learning Research, 17(1), 1235-1241.

[12] Shoro, A. G., & Soomro, T. R. (2015). Big data analysis: Apache spark perspective. Global Journal of Computer Science and Technology.

[13] Hows, D., Membrey, P., Plugge, E., & Hawkins, T. (2015). Introduction to mongodb. In The Definitive Guide to MongoDB (pp. 1-16). Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-3052-6_1

[14] Boicea, A., Radulescu, F., & Agapin, L. I. (2012). MongoDB vs Oracle--database comparison. In 2012 third international conference on emerging intelligent data and web technologies (pp. 330-335). IEEE. https://doi.org/10.1109/EIDWT.2012.32

[15] Zhao, G., Huang, W., Liang, S., & Tang, Y. (2013). Modeling MongoDB with relational model. In 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies (pp. 115-121). IEEE. https://doi.org/10.1109/EIDWT.2013.25