International Journal of Engineering Science and Applied Mathematics
Volume.13, Number 3; March-2022;
ISSN: 2836-9521| Impact Factor: 5.78
https://zapjournals.com/Journals/index.php/ijesam
Published By: Zendo Academic Publishing

Strengthening Privacy Measures in Data Publishing: Exploring the Bit-Coded-Sensitive Algorithm (BCSA) Approach

[1]Z. Mohamad Said and [2]M. F. Zolkipli

Article Info

Keywords: privacy-preserving data publishing, PPDP, k-anonymity, data anonymization, Bit-coded-sensitive algorithm, privacy protection, data mining, and privacy threats, confidentiality, information patterns.

Abstract

Privacy-preserving data publishing (PPDP) is a crucial field of study that aims to safeguard individual privacy while enabling the release of useful information for data mining purposes. The widespread availability and exchange of large volumes of data in organizations necessitate the extraction of hidden information patterns for decision-making processes. However, this practice poses challenges to privacy and confidentiality. When organizations publish substantial amounts of data, there is a risk of exposing sensitive information, resulting in privacy violations. Therefore, it is imperative to protect information and restrict access to authorized individuals. This paper addresses the limitations of existing methods for achieving PPDP by proposing a novel technique called the "Bit-coded-sensitive algorithm" (BCSA).

Existing approaches, such as k-anonymity, aim to anonymize datasets by removing key attributes that can be linked to individuals. However, simply removing attributes does not guarantee anonymity. The BCSA algorithm overcomes this limitation by efficiently and effectively preserving individual privacy. It can be combined with other algorithms, enhancing its resistance against adversarial attacks. This paper presents a comprehensive framework that encompasses k-anonymity in publishing data, previous work, the problem definition, the BCSA algorithm, and a conclusion.

The proposed BCSA algorithm addresses the drawbacks of other anonymization methods, including grouping-and-breaking, suppression, generalization, bucketization, and perturbation. It ensures that membership disclosure is prevented, establishes a clear distinction between quasi and sensitive attributes, and maintains the link between these attributes. By incorporating the BCSA algorithm, organizations can achieve better data utility while upholding privacy requirements.

[1] Koforidua Technical University, Koforidua, Ghana
[2] University of Cape Coast, Cape Coast, Ghana

The presented framework offers a valuable contribution to the field of PPDP and serves as a foundation for future research and advancements

1 Introduction

Privacy persevering data publishing (PPDP) is a study that seeks to remove privacy threats and ensure useful information is released data for mining. As a result of the vast volumes of data that flow in and out of organizations, the concept of mining is used to extract hidden information patterns from large data sets for decision-making processes. This though important is faced with challenges such as privacy and confidentiality of the individual. Thus, when organizations publish a massive amount of data they collect, there inevitably arises the case that sensitive pieces of information are exposed. This leads to the violation of privacy [1] and it is critical to guarantee that information is protected and accessible only to authorized individuals [2].

Therefore, to ensure data safety and integrity, providers of data try to remove vital features of data sets that can be associated with an individual. These include the name, address, SSN No, and ID. The removal of key attributes however does not guarantee the anonymity of an individual's data. K-anonymity is a method that has been proposed to achieve PPDP. It ensures that individual tuples found in the published microdata cannot be distinguished from at least k-1 other tuples. K-anonymity follows two main methods for anonymizing tables before publication. First is grouping-and- breaking. This method divides a record into a set of partitions in a horizontal manner. Once this is done all links between quasi and sensitive attributes in the various partitions are broken. This creates a set of individual attributes whose attributes cannot be related to any particular individual. An attacker would find it difficult to create inferences using quasi-identifier values and sensitive values [3][4][5]. Suppression is the removal of an entire tuple or record or attribute value. This is then replaced with a special symbol such as (*). The disadvantage of suppression is that it reduces the quality of data. Generalization replaces attributes with semantically consistent but less specific values. The limitations of generalization include loss of information for high dimensional data, decrease in data utility, and loss of correlation between different attributes. Bucketization is also used in k-anonymity. It does not modify quasi-identifiers or sensitive identifiers. It splits records into partitions and gives each partition an ID known as GID. Thus, all tuples in the record will have the same GID. This process produces two tables. One table consists of quasi-attributes while the other has sensitive attributes. The disadvantages of the above methods are that it fails to prevent membership disclosure, do not give a clear difference between quasi and sensitive attributes, and the link between quasi and sensitive attributes is broken just as with generalization. The other way of anonymizing the table is by perturbation. In this method, values can be replaced with other ones. For example, a male can be switched to a female and vice versa. The drawback of this is that it reduces the utility of data.

This paper proposes a technique to address the above-mentioned shortfalls known as the "Bit-coded-sensitive algorithm'' BCSA. This algorithm is more efficient and effective and ensures that the privacy of the individual is preserved. In addition, it can be used with other algorithms making it more difficult for an adversary. The paper is hereafter as follows section (ii) is on K-anonymity in publishing data, section (iii) is on previous work, and section (iv) is the problem definition. Section (v) presents the framework. Section (vi) is the conclusion. A list of references is presented in section (vii).

2 K-anonymity in publishing data

K-anonymity ensures that each tuple in the published microdata such as data from the census cannot be distinguished from at least k−1 other tuples [6][7][8]. K-anonymity consists of three types of attributes. The most important attribute is the Key Attribute. The name and identification number of an individual can be classified as Key Attributes. Before microdata is released, Key Attributes are usually removed. The Quasi-Attribute is the next

type of attribute used in k-anonymity. These include gender, age, nationality, and zip code. These are general attributes that can be found in public databases. They are used as means of re-identifying individuals. Last but not the least, the last attribute is the sensitive attribute. These attributes are usually published directly. Tables 1 and 2 show a classification of the various attributes and the distinction between sensitive and non-sensitive attributes used in publishing microdata respectively of people who visited a hospital.

Table 1. Attribute classification in k-anonymity

Key Attributes	**Quasi er Identifi**			**Sensitive Attributes**
NAME	GENDER	AGE	ZIP CODE	DISEASE
JOHN	FEMALE	25	423101	FLU
JOANA	MALE	27	423308	HIV+

Table 2. Distinction between sensitive and non-sensitive attributes

S.No	**Attributes**	**Type**
1	ZIP CODE	NON-SENSITIVE
2	AGE	NON-SENSITIVE
3	NATIONALITY	NON-SENSITIVE
4	MEDICAL STATUS	SENSITIVE
5	OCCUPATION	SENSITIVE
6	ANNUAL INCOME	SENSITIVE

Table 3. Original information

Name	**Gender**	**Zip Code**	**Age**	**Disease**
KOFI	M	431201	40	HEART DISEASE
YAW	M	444806	23	FLU
KYEI	M	444601	26	HEART DISEASE
AFIA	F	431001	35	FLU
KWASI	M	431203	38	VIRAL INFECTION
AMA	F	444694	20	CANCER

Table 4. The use of suppression on original data

Gender	**Zip Code**	**Age**	**Disease**
*	431201	*	HEART DISEASE
*	444806	*	FLU
*	444601	*	HEART DISEASE
*	431001	*	FLU

*	431203	*	VIRAL INFECTION
*	444694	*	CANCEL

Table 5. The use of generalization on original data

Zip Code	Gender	Age	Disease
431201	PERSON	35–40	HEART DISEASE
444806	PERSON	20–26	FLU
444601	M	20–26	HEART DISEASE
431001	PERSON	35–40	FLU
431203	F	35–40	VIRAL INFECTION
444694	PERSON	20–26	CANCER

Table 6A. The use of bucketization on original data

Gender	Zip Code	Age	Disease	GID
F	431201	40	HEART DISEASE	1
F	444806	23	FLU	1
M	444601	26	HEART DISEASE	2
M	431001	35	FLU	2
F	431203	38	VIRAL INFECTION	3
F	444694	20	CANCER	3

Table 6B.

GID	Disease
1	HEART DISEASE
1	FLU
2	HEART DISEASE
2	FLU
3	VIRAL INFECTION
3	CANCER

Table 7. Use of perturbation on original data

Gender	Zip Code	Age	Disease
F	431201	40	HEART DISEASE
F	444806	23	FLU
M	444601	26	HEART DISEASE

M	431001	35	FLU
F	431203	38	VIRAL INFECTION
F	444694	20	CANCER

Table 8. Original data from Hospital

ID	**Age**	**Country**	**Zip Code**	**Disease**
1	27	USA	14248	HIV
2	28	Canada	14207	HIV
3	26	USA	14206	Cancer
4	25	Canada	14249	Cancer
5	41	China	13053	Hepatitis
6	48	Japan	13074	Phthisis
7	45	India	13064	Asthma
8	42	India	13062	Obesity
9	33	USA	14242	Flu
10	37	Canada	14204	Flu
11	36	Canada	14205	Flu
12	35	USA	14248	Indigestion

3 Previous work

Data Mining which is done to preserve data typically involves the use of various techniques to change the original information set or the one that is generated after data has been mined [9]. To ensure efficient results during this process and still preserve the integrity of data that belongs to people. Five main things are considered in the process of ensuring data privacy. These are

i. Distribution
ii. Modification
iii. Mining Method iv. Hiding of basic rules or data
v. Additional methods for the preservation of privacy [10].

Thus, lots of different methods and techniques have been discovered to be used in the context of PPDM.

3.1 Cryptographic approach

In [11] the authors introduced the cryptographic approach. This approach has two objectives. The first is to define a suitable model for privacy. The second is to implement an algorithm that facilitates the preservation of data that has been mined. There are many cryptographic algorithms and approaches which meet the above criteria, however recent work indicates that this approach prevents the leakage of data during computation but does not protect the outcome of the computation process. Also, the cryptographic method cannot be used for smaller data sets. Finally, this method fails to provide an answer to the significant question of whether exposure to results after data mining can lead to a privacy breach of the individual.

3.2 Blocking based method

In this method, the 1's or 0's in selected transactions are replaced by unknowns ("?"). This ensures that rules are not generated from the dataset. The algorithm has the sole aim of hiding any given set of sensitive rules by

swapping known values with ones that cannot be deciphered. When the algorithm identifies a sensitive rule, it scans the original database for transactions supporting that rule when it identifies a sensitive rule [12].

3.3 Condensation approach

Another approach was introduced by [4][13] and is known as condensation. In their approach, a cluster of data set is generated, and a pseudo-code is generated using statistics for the clusters. Condensation is then used to compress the data into multiple groups with the predefined group each with a size of at least K. This however suffers from information loss.

3.4 Random perturbation approach

In this method, the privacy of data is protected by using a randomization algorithm to alter sensitive data before a data miner has access to it [14][15]. The original data is therefore distorted through the addition of noise. This helps to solve the problem of character and classification types, Boolean and number types associated with [16] deal with character type, classification type as well as the Boolean type and number type associated with discrete data. The facilitation of data set conversion is done through the pre-processing of original data. [17] uses the method of the average region to disperse the continuous data. Discrete formula is as follows: A (max) – A (min)/n = length. A is a continuous attribute, n is a discrete number, and length is the length of discrete intervals. When the interval length is a decimal, round to the nearest integer, the first interval of discrete begins from A (min), and the last interval is A (max).

4 Problem definition

The question we try to answer is, must sensitive attributes be published for any reason, and if so, how must they be published? Sensitive attributes such as disease, credit card numbers, social security numbers, occupation, income, and bank account numbers are very personal and significant. As a result, these must not be disclosed no matter what. There have been several instances where people have stolen for example other people's credit cards and have gained access to their account numbers for the wrong reasons. Because of this, we propose a model known as the Bit-Coded Sensitivity Algorithm (BCSA). The model identifies sensitive data and then encrypts these sensitive data before it is published. The unique thing about this model is that it can be deployed together with other algorithms. Unlike other models, this model ensures that sensitive attributes are not disclosed and hence prevents reidentification, linkages, and information loss while increasing data utility. If microdata contains sensitive attributes, then sensitive attributes must be such that it is unknown and not understood by an unauthorized individual.

5 Methodology

5.1 Bit-Coded-Sensitive Algorithm (BSCA)

This a basic implementation of BCSA which encrypts based on a key given. This key should be kept private and must be unique for each cipher.

5.2 Implementation

Steps to encrypt

– encrypt method (string, key) – using this as our list of alphabetic characters cipher_ blocks = [‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’, ‘J’, ‘K’, ‘L’, ‘M’, ‘N’, ‘O’, ‘P’, ‘Q’, ‘R’, ‘S’, ‘T’, ‘U’, ‘V’, ‘W’, ‘X’, ‘Y’, ‘Z’, ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ‘h’, ‘i’, ‘j’, ‘k’, ‘l’, ‘m’, ‘n’, ‘o’, ‘p’, ‘q’, ‘r’, ‘s’, ‘t’, ‘u’, ‘v’, ‘w’, ‘x’, ‘y’, ‘z’, ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’].

– encrypt method takes in plain text or strings as the first argument and the key as the second argument. Encrypt (plaintext, key).

– using the length of the plain text or strings as the highest number of the iteration process. for i in range(len(plaintext)).

– checks if ord(letter) == 32 which 32 is a space value in unicode. it skips the iteration or jumps the iteration.
– Get the index of each letter from the block list index=cipher_blocks.index(letter).
– Now to ciphering, adds the key to the index modulus 62 to get a new char which is the appended to ciphertext += cipher_blocks[(index+key) % 62]
– checks if letter is numeric letter.isnumeric() then get the index of the letter from the block list index=cipher_blocks.index(letter).
– Now to ciphering, add the key to the index modulus 62 to get a new char which is then appended to ciphertext += cipher_blocks[(index+key) % 62.
– return ciphertext as the loop exit return ciphertext.

Steps to decrypt

Decrypt (ciphertext, key) – using this as our list of alphabetic characters cipher_ blocks = [‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’, ‘J’, ‘K’, ‘L’, ‘M’, ‘N’, ‘O’, ‘P’, ‘Q’, ‘R’, ‘S’, ‘T’, ‘U’, ‘V’, ‘W’, ‘X’, ‘Y’, ‘Z’, ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ‘h’, ‘i’, ‘j’, ‘k’, ‘l’, ‘m’, ‘n’, ‘o’, ‘p’, ‘q’, ‘r’, ‘s’, ‘t’, ‘u’, ‘v’, ‘w’, ‘x’, ‘y’, ‘z’, ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’].
– decrypt method takes in encrypt text or strings as the first argument and the key as the second argument. decrypt (encrypt text, key).
– using the length of the encrypted text or strings as the highest number of the iteration process. for i in range(len(ciphertext)).
– initialize letter to index of ciphertext based on the current value of loop letter = ciphertext[i]
– searches through cipher_blocks and pick the index of the letter from previous step index = cipher_blocks.index(letter) .
– Now to ciphering, subtract the key to the index modulus 62 to get an original char which is then appended to the plaintext += cipher_blocks[(index-key) % 62].
– returns plaintext when the loop exit.

5.3 Description of the proposed algorithm

Encryption. Using a set of chars in an array or a list that consists of letters from A to Z and a–z and numbers from 0 to 9. We first start by getting the Unicode of a specified character ignoring a space character (i.e., space). The function that encrypts, takes plain text and a secret key (a number) which is used for the encryption. The function starts by getting the length of the plain text and using that as the number of times to literate. Thus, in a loop, it gets the unicode value of a character and also checks if it is not equal to 32 which is a space. If the Unicode value is 32, it skips the loop and moves to the next character.

Using the secret key given, it adds the secret key to the Unicode value of the character to get a new character by mapping it to the array of characters. However, to make sure the addition of the character in Unicode and the secret key does not exceed 62 (which is the total length of characters from A–Z, a–z, and 0–9), the sum is divided by the 62 and only the remainder is returned. This happens several times per the depth of the loop till all plain text is encrypted.

Decryption. Decryption only works if the ciphered text was encrypted with the encryption algorithm above.

Using a set of chars in an array or a list that consists of letters from A to Z, a–z, and numbers from 0 to 9. We first start by getting the Unicode of a specified character, ignoring the space character (i.e., space). The function that decrypts takes the encrypted text and a secret key (a number) which is used for the decryption. The function starts by getting the length of the encrypted text and using that as the number of times to iterate. Thus, the loop gets the unicode value of a character and also checks if not equal to 32 which is a space. If the unicode value is 32 which is a space it skips the loop and moves to the next character. Using the secret key given, it subtracts the secret key

from the Unicode value of the character. However, to make sure the addition of the character in Unicode and the secret key does not exceed 62 (which is the total length of characters from A–Z, a–z, and 0–9), the sum is divided by the 62 but only the remainder is returned to get a new character by mapping it to the array of characters. This happens several times per the depth of the loop till all encrypted text is decrypted.

6 Analysis

Most industry in the world collects and stores huge amounts of sensitive data on computers and other digital devices. These sensitive data often include personal data to intellectual properties, government documents, medical records, and bank details which when intercepted or accessed by unauthorized individuals, often have negative repercussions. Protecting data or information require three main certain security features that are of great concern to most businesses and organization. Upon experimenting with K-anonymity algorithms such as suppression, generalization, and buckertization, it was realized that when an individual knows the source of published data, it is easy to have an idea about sensitive data and hence establish linkages through trial and error or guesses. This leads to the disclosure of sensitive data. Additionally, hackers and cybercriminals, most of the time utilize this approach to attack critical systems. The impact of these attacks has an undying effect on Individuals, small-scale businesses, and large organizations. The proposed algorithm BCSA is to ensure that sensitive data and information are not available and accessible to unauthorized individuals. This algorithm is expected to be more efficient and effective and can be used with other existing algorithms.

The following assumptions were made for the experiment. Data were classified into two main categories:

Sensitive data: This includes

- Name
- Account details, social security number, income, occupation.
- Diseases that can cause stigmatization or death: HIV, Cancer, Coronavirus
- Criminal History: Rape, other forms of abuse

The following are considered highly sensitive. *Quasi-attributes*

- Age
- Zip code
- Nationality

These are considered low-sensitive data and as such cannot cause harm if published by an adversary. This is because in most cases it can be hypothetical.

The authors recognize the significance of sensitive data or information and how it can be used to defame or cause a distraction to an individual. For example, diseases such as HIV, heart disease, cancer, and accounts details. On their own, these data are of no significance as they cannot be linked to any individual. However, when key attributes and quasi-attributes are also published in the released dataset (also known as masked microdata and labeled *MM*) without the application of the correct privacy-preserving technique, algorithm, or tool, then this can present an opportunity for an adversary to take advantage. Consequently, this can lead to unauthorized disclosure or linkage that can cause harm to the individual concerned.

To address the challenges posed by the key anonymity and other privacy-preserving data algorithms, we identify the source or origin of the initial dataset, (known as initial microdata and labeled *IM*) and placed it in three categories namely Health, Bank, and Transactional or E-commerce types. These categories of data are considered very sensitive. We use an algorithm known as the bit-coded-sensitive algorithms (BCSA) to code each of the three categorized sources of data. Thus, sensitive data after mining are coded with a key only known by the data miner. The encrypted sensitive data is then published without the key in the masked microdata as shown in Table 9. In this way, only authorized individuals have the key or know the key to decrypt a record with sensitive data.

However, in the case where for instance, a disease occurs more than once in a patient's records, different codes can be generated for the same disease using the BCSA and hence different keys to decrypt.

Table 9. Using BCSA on sensitive data on Table 8

ID	Age	Country	Zip Code	Disease
1	27	USA	14248	OIE
2	28	Canada	14207	HFSHJW
3	26	USA	14206	15QKT
4	25	Canada	14249	VXZDRFP
5	41	China	13053	CFLUOH
6	48	Japan	13074	27RHUP
7	45	India	13064	357FRZ
8	42	India	13062	SBJ56TY
9	33	USA	14242	13OIFTBN
10	37	Canada	14204	FI06
11	36	Canada	14205	TMCXWQ
12	35	USA	14248	JTFWXAD

Table 10. Published data using BCSA together with other algorithms on Table 8

ID	Age	Country	Zip Code	Disease
1	(27–29)	America	142**	OIE
2	(27–28)	America	142**	HFSHJW
3	(25–26)	America	142**	15QKT
4	(25–26)	America	142**	VXZDRFP
5	>41	Asia	130**	CFLUOH
6	>48	Asia	130**	27RHUP
7	>45	Asia	130**	357FRZ
8	>42	Asia	130**	SBJ56TY
9	(33–35)	America	142**	13OIFTBN
10	(36–37)	America	142**	FI06
11	(36–37)	America	142**	TMCXWQ
12	(33–35)	America	142**	JTFWXAD

The Bit-Coded Sensitive algorithm defines a cipher block containing uppercase, and lowercase alphabets, and numerals. The algorithm goes through the plaintext and checks if there is a letter that is an alphabet or a number. For uppercase values, the plaintext is converted into a ciphertext consisting based on modulo (62) of an index (uppercase) and key value. For numeric values, the plaintext is converted into a ciphertext consisting based on modulo (62) of an index(lowercase). The algorithm then returns the ciphertext.

A decryption algorithm is also defined which decrypts the ciphertext and allows one to obtain the plaintext.

7 Discussion of results

In Table 9, the BCSA is applied to code the sensitive data in Table 8. This makes it difficult for an unauthorized person to understand the whole table after it is published since only authorized individuals have the key or know the key to decrypt a record with sensitive data. In the case where for instance, a disease occurs more than once in a patient's records, different codes can be generated for the same disease using the BCSA and hence different keys to decrypt.

In a test conducted after the publishing of microdata as shown in Table 9, ten (10) students were asked to interpret and use the available data after briefing students on the source of data. Students were unable to understand the microdata and therefore could not deduce any significant information from the published microdata. However, with the use of suppression, generalization, and bucketization in Tables 4–6 respectively, about 50% of these students through guesses were able to establish linkages and also disclose information about individuals in the original data. The use of BCSA prevents reidentification, linkages, and information loss while increasing data utility by authorized individuals. If microdata contains sensitive attributes, then sensitive attributes must be such that it is unknown and not understood by an unauthorized individual. To enhance the effectiveness of BCSA, we combined it with other algorithms such as suppression, generalization, and bucketization and realized that BCSA is very effective as compared to just using these algorithms on their own.

Using BCSA is very efficient as it requires keys to be able to understand and use published data effectively in decision-making in every organization or institution. It prevents and denies unauthorized users access to sensitive data or information while at the same time offering protection and security to data and individuals.

8 Conclusion

From the test carried out, it was realized that using the BCSA, makes it difficult for an adversary to identify or disclose information about an individual or perform linkages since he does not have the key that was used to code sensitive data. Additionally, information is not lost and data can be highly utilized by authorized users. Furthermore, when BCSA is compared with generalization, suppression, and buckertization, it was revealed that BCSA is very efficient, effective, and more secure.

9 References

A. T. Ravi and S. Chitra, "Privacy-preserving data mining," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 9, no. 8, pp. 616–621, 2015. https://doi.org/ 10.19026/rjaset.9.1445

Z. Mohamad Said and M. F. Zolkipli, "Internet of Things (IoT): A study of security issues and challenges," *International Journal of Recent Contributions from Engineering, Science & IT (iJES)*, vol. 10, no. 2, pp. 16–31, 2022. https://doi.org/10.3991/ijes.v10i02.29301

J. Wang, Y. Luo, Y. Zhao, and J. Le, "A survey on privacy preserving data mining," *Proceedings – 2009 1st International Workshop on Database Technology and Applications, DBTA 2009*, pp. 111–114, 2009. https://doi.org/10.1109/DBTA.2009.147

V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *SIGMOD Record*, vol. 33, no. 1, pp. 50–57, 2004. https://doi.org/10.1145/974121.974131

L. Sweeney, "A model for protecting privacy," *IEEE S&P '02*, vol. 10, no. 5, pp. 1–14, 2002.

S. Vijayarani, A. Tamilarasi, and M. Sampoorna, "Analysis of privacy-preserving K-anonymity methods and techniques," *Proceedings of 2010 International Conference on Communication and Computational Intelligence, INCOCCI-2010*, pp. 540–545, 2010.

K. Wang, P. S. Yu, and S. Chakraborty, "Bottom-up generalization: A data mining solution to privacy protection," *Proceedings – Fourth IEEE International Conference on Data Mining, ICDM 2004*, pp. 249–256, 2004.

B. C. M. Fung, K. Wang, and P. S. Yu, "Top-down specialization for information and privacy preservation," *Proceedings – International Conference on Data Engineering*, no. Icde, pp. 205–216, 2005.

E. Poovammal and M. Ponnavaikko, "Task independent privacy preserving data mining on medical dataset," *ACT 2009 – International Conference on Advances in Computing,*

Control and Telecommunication Technologies, pp. 814–818, 2009. https://doi.org/10.1109/ ACT.2009.206

M. Sharma, A. Chaudhary, M. Mathuria, S. Chaudhary, and S. Kumar, "An efficient approach for privacy preserving in data mining," *2014 International Conference on Signal Propagation and Computer Technology, ICSPCT 2014*, pp. 244–249, 2014. https://doi.org/10.1109/ ICSPCT.2014.6885001

K. Chen and L. Liu, "Privacy preserving data classification with rotation perturbation," *Proceedings – IEEE International Conference on Data Mining, ICDM*, pp. 589–592, 2005.

J. Liu, J. Luo, and J. Z. Huang, "Rating: Privacy preservation for multiple attributes with different sensitivity requirements," *Proceedings – IEEE International Conference on Data Mining, ICDM*, pp. 666–673, 2011. https://doi.org/10.1109/ICDMW.2011.144

A. A. Parmar, U. P. Rao, and D. R. Patel, "Blocking based approach for classification rule hiding to preserve the privacy in database," *Proceedings – 2011 International Symposium on Computer Science and Society, ISCCS 2011*, pp. 323–326, 2011. https://doi.org/10.1109/ ISCCS.2011.103

C. C. Aggarwal and P. S. Yu, "A condensation approach to privacy preserving data mining," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2992, no. June 2014, pp. 183–199, 2004. https://doi.org/10.1007/978-3-540-24741-8_12

H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "Random-data perturbation techniques and privacy-preserving data mining," *Knowledge and Information Systems*, vol. 7, no. 4, pp. 387–414, 2005. https://doi.org/10.1007/s10115-004-0173-6

X. Zhang and H. Bi, "Research on privacy preserving classification data mining based on random perturbation," *ICINA 2010–2010 International Conference on Information, Networking and Automation, Proceedings*, vol. 1, pp. 1–6, 2010.

D. A. Kumari, K. R. Rao, and M. Suman, "Privacy preserving data mining," *Advances in Intelligent Systems and Computing*, vol. 249, pp. 517–524, 2014. https://doi.org/10.1007/ 978-3-319-03095-1_55