Design and Implementation of Reversible Database Watermarking

Ruiqi Wang*

Xinjiang University, Urumqi, China

* 3204531075@qq.com

Abstract

With the rapid development of information technology, databases are increasingly facing security threats such as data leakage, tampering, and unauthorized replication, which traditional database protection methods are now struggling to counter. Database watermarking technology is an important means of data security. It can address copyright protection and data tracking to a certain extent. However, it still faces challenges in balancing watermark robustness, data distortion, and privacy risks. In response to these issues, this paper proposes a database watermarking algorithm based on differential expansion technology and the principle of reversibility, and delves into the application of the ChaCha20 stream cipher algorithm in watermark information encryption. The proposed database watermarking algorithm analyses database characteristics to determine embedding positions and intensity, achieving effective watermark embedding with minimal impact on data integrity. The ChaCha20 algorithm enhances watermark concealment by generating a pseudo-random keystream for encryption, improving robustness against data processing. Experimental validation shows that the algorithm proposed in this paper is reversible and robust, capable of protecting database copyrights and tracing data origins while ensuring data integrity and availability. By combining differential expansion and ChaCha20 encryption, this approach provides an efficient and reliable technical means for database security protection. This research offers new perspectives for database watermarking technology and contributes to further development and application in the field.

Keywords

Reversible database watermarking, differential expansion, data recovery, hash function, ChaCha20.

1. Introduction

The research trajectory of reversible database watermarking dates back to the early development of digital watermarking technology. In the early 1990s, digital watermarking technology, as a form of information hiding, began to garner attention from researchers. In 1994, at the IEEE International Conference on Image Processing, Schyndel R.G. and colleagues first introduced the concept of 'digital watermarking' and discussed an 'undetectable' digital watermarking technique based on the least significant bit (LSB) algorithm for standard image coding. Research during this period primarily concentrated on image processing and had not yet extended to the database field[1]. In 1997, American scholar Barton first proposed the concept of reversible digital watermarking algorithms. This algorithm established the theoretical foundation for the subsequent development of reversible database watermarking technology. Entering the 21st century, with the rapid advancement of database technology, research on reversible database watermarking technology has progressively deepened. Researchers began designing various algorithms aimed at embedding and extracting

watermark information while maintaining data integrity. In 2002, Agrawal [2] first proposed the application of digital watermarking technology to relational databases for database copyright protection. Although this algorithm is classified as irreversible watermarking, it provided a significant reference for subsequent research in reversible database watermarking technology.

Building upon these early foundations, researchers continuously refined algorithms to improve the performance of reversible database watermarking technology. For instance, by optimizing embedding strategies and enhancing watermark robustness, reversible database watermarking technology can more effectively withstand various attacks and tampering. Di Guandong, Chai Heyan [3,4] and colleagues realized more robust reversible database watermarking by introducing an improved clustering algorithm, whereas Xie Jingsong, Jiang Chuanxian [5,6] and others proposed an implementation scheme based on the simulated annealing algorithm. This method utilizes simulated annealing to optimize the modification magnitude of attribute column data to be embedded, thereby improving the watermark's imperceptibility. Jiang Chuanxian [7] A reversible database watermarking algorithm based on integer wavelet transform was also proposed. This algorithm analyzes the high-frequency coefficients in the wavelet domain of the database data and embeds the watermark into this domain to enhance watermark imperceptibility. Duan Jiangbing and Jiang Chuanxian[8,9]and colleagues, leveraging the two key characteristics of relational data-low redundancy and high discreteness-proposed a reversible database watermarking algorithm based on discrete particle swarm optimization. Long Xiaoqian's research effectively guarantees database integrity and security by integrating support vector regression with frequent pattern tree techniques[10]. Additionally, Li Yue and Yang Canji have also explored integrating reversible database watermarking technology with other database security techniques, such as order-preserving encryption and public-private key encryption, to provide a more comprehensive database protection scheme[11,12]. Research on reversible database watermarking technology remains in a continuous process of development and refinement. With the advent of new technologies and the expansion of application scenarios, reversible database watermarking technology is expected to play an increasingly important role in the future, providing strong support for database security protection[13].

2. Watermark Encryption Technology based on ChaCha20

2.1. ChaCha20 Algorithm

The ChaCha20 algorithm is an improved stream cipher derived from Salsa20, designed by cryptography expert Daniel J. Bernstein. It inherits the excellent characteristics of Salsa20 and further optimizes both security and performance. ChaCha20 is designed to offer enhanced security while maintaining or improving the algorithm's execution efficiency. This algorithm has been adopted as an IETF standard and is widely used in various security protocols and sensitive data protection.

2.1.1. Overall Process

ChaCha20 is a stream cipher that generates a pseudorandom keystream, which is then XORed with the watermark plaintext information to produce the ciphertext. In each encryption, a distinct nonce is used along with the same key; this ensures that even if an identical plaintext is encrypted multiple times, each resulting ciphertext is unique, thereby enhancing the security and diversity of the encryption. The decryption process is the inverse of encryption: using the same key and nonce, the ciphertext is XORed with the identical keystream to recover the original watermark plaintext information. The following describes the process whereby ChaCha20 generates the keystream and encrypts the plaintext:

1) Initialization

First, a key and a nonce (also referred to as an initialization vector, IV) are used to initialize the ChaCha20 state matrix. The state matrix is a 4x4 matrix comprising the key, nonce, and fixed constants. The key may be user-provided, generated via a key derivation function, or randomly produced for each encryption. The key length is typically 128 bits, the standard size for the ChaCha20 algorithm, which provides adequate security while ensuring good performance. The state matrix is illustrated in Figure 1:

constant	constant	constant	constant
key	key	key	key
key	key	key	key
nonce	nonce	nonce	nonce

Figure 1. Initial state matrix.

2) Keystream generation

In the ChaCha20 algorithm, the process of keystream generation constitutes the core of the encryption system. This process involves a series of meticulously designed encryption rounds, each comprising a set of complex mathematical operations intended to enhance data security through diffusion and confusion mechanisms. Specifically, the algorithm performs 20 rounds of computation, each divided into four quarter rounds, with each quarter round consisting of a sequence of fundamental operations applied to the algorithm's initial state matrix. In each round, every word of the state matrix is updated according to a predetermined pattern. The addition operation provides the fundamental nonlinear transformation, the exclusive OR operation introduces mixing and complexity, and the bit rotation operation further scrambles the data bits, enhancing the algorithm's diffusion. These operations work synergistically to ensure that each word extracted from the state matrix is influenced by the state from the previous round, thereby making the final generated keystream highly random and unpredictable[14].

3) Encrypting the watermark plaintext information

In the ChaCha20 algorithm, the generated keystream is combined with the predefined watermark plaintext information through an exclusive OR operation to achieve encryption. Specifically, this process performs a byte-wise exclusive OR operation between each byte of the keystream and the corresponding byte of the watermark plaintext information. Through this operation, every byte of the watermark plaintext information is converted into its corresponding ciphertext byte, generating the encrypted watermark ciphertext. Overall, ChaCha20 does not merely XOR the plaintext with a fixed value; rather, it generates a keystream via a complex, key- and nonce-dependent algorithmic process, which is then used to encrypt the plaintext. This approach provides strong security, enabling the ciphertext to be transmitted securely, as only those possessing the correct key and nonce can decrypt it accurately.

2.1.2. State Matrix

In the ChaCha20 algorithm, the state matrix undergoes 20 rounds of carefully designed encryption operations, which can be divided into two fundamental types: column operations and diagonal operations. Specifically, the first round concentrates on column operations, applying quarter rounds to each column in parallel. This process starts from the second row, then sequentially processes the third row, the fourth row, and the first row. If matrix rows are stored as vectors, these four quarter rounds can be fully implemented via single instruction multiple data (SIMD) operations on vectors, thus improving computational efficiency. The second round focuses on the diagonal operation, processing the northeast diagonal in the order of the fourth row, third row, second row, and first row. By appropriately rotating the entries

within each row, the diagonal operation can be converted into a column operation, enabling the use of SIMD operations to reduce computational cost while preserving the algorithm's parallelism. From the third round onward, ChaCha20's operation mode repeats the patterns of the first and second rounds. Specifically, the third round resembles the first round, focusing on column operations; the fourth round resembles the second round, focusing on diagonal operations. This cyclical pattern continues until the completion of all 20 rounds. By systematically combining column and diagonal operations, the ChaCha20 algorithm not only ensures thorough data mixing but also optimizes computational efficiency, enabling effective execution on various hardware platforms. Upon completion of each operational round, the state matrix undergoes new transformations; these changes accumulate to produce a highly random keystream, thereby providing strong security guarantees for the data[15].

3. Reversible Database Watermarking Method based on Differential Expansion

3.1. Differential Expansion Technique

In reversible database watermarking technology, the application of differential expansion is primarily based on the numerical relationships of data pairs (such as adjacent records or correlated fields in the database). Suppose the attribute value for watermark embedding is A_1, A_2 , and the watermark bit to be embedded is b. The formulas for calculating the mean value, avg, and the difference d between two attribute values are shown in (1):

$$\begin{cases} avg=\left\lfloor \frac{A_1+A_2}{2} \right\rfloor \\ d=A_1-A_2 \end{cases} \tag{1}$$

During the watermark embedding phase, the watermark is embedded by expanding the difference; specifically, the difference d is adjusted according to the watermark bit b, as shown in formula (2):

$$d'=2\times d+b \tag{2}$$

The attribute values embedded with the watermark are calculated according to formulas (3) and (4).A_1',A_2':

$$\begin{cases} g=\left\lfloor \frac{d'+1}{2} \right\rfloor \\ f=\left\lfloor \frac{d'}{2} \right\rfloor \end{cases} \tag{3}$$

$$\begin{cases} A_1'=avg+g \\ A_2'=avg-f \end{cases} \tag{4}$$

During watermark extraction and data restoration, the difference between the data pairs embedded with the watermark is recalculated.d' and the mean value, avg, as shown in formulas (5):

$$\begin{cases} avg'=\left\lfloor \frac{A_1'+A_2'}{2} \right\rfloor \\ d'=A_1'-A_2' \end{cases} \tag{5}$$

Then, the watermark bit b is extracted by utilizing the difference between attribute values. As shown in formulas (6) and (7):

$$d'=\lfloor d'/2 \rfloor \tag{6}$$

$$b=d'-2\times d' \tag{7}$$

Finally, the original A1 and A2 attribute values are restored through avg' and d', as shown in formulas (8) and (9):

$$\begin{cases} g=\left\lfloor \frac{d'+1}{2} \right\rfloor \\ f=\left\lfloor \frac{d'}{2} \right\rfloor \end{cases} \tag{8}$$

$$\begin{cases} A_1=avg'+g \\ A_2=avg'-f \end{cases} \tag{9}$$

The advantage of the differential expansion technique lies in its ability to embed watermark information while maintaining data availability and to perfectly restore the original data through inverse operations. This endows it with broad application prospects in fields such as database watermarking. However, the differential expansion technique requires balancing data distortion and watermark robustness to ensure optimal performance in practical applications.

3.2. Detailed Description of the Algorithm Process

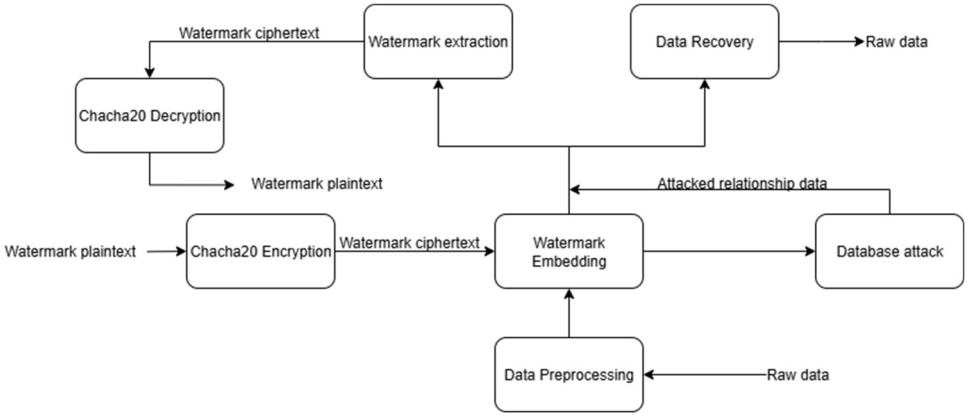

Figure 2. Implementation process of the reversible database watermarking algorithm.

As shown in Figure 2, the reversible database watermarking method based on differential expansion primarily comprises four stages: data preprocessing, watermark embedding, watermark extraction, and data recovery. Data preprocessing is the first step of the reversible database watermarking method, aiming to select data tuples suitable for watermark embedding from the raw database and to adequately prepare data for the subsequent watermark embedding process. Watermark embedding is the core component of the reversible database watermarking method. It selects the least significant eight bits of the data as the watermark embedding attribute bits and embeds the watermark sequence into these bits using differential expansion transformation. Watermark extraction and data recovery constitute the final step of the reversible database watermarking method. This step extracts the watermark

sequence and restores the original database by performing the inverse differential expansion operation on the watermark-embedded database.

3.2.1. Data Preprocessing

The data preprocessing stage is the initial step of the reversible database watermarking algorithm. Its primary objective is to select data tuples suitable for watermark embedding and to prepare for the subsequent watermark embedding process. The specific steps of data preprocessing are as follows:

1) Generate a unique identifier: Perform an MD5 hash on the unique ID of each data tuple to produce a fixed-length hash value. This step not only ensures the uniqueness of data tuples but also provides the necessary preparation for the subsequent watermark embedding process.

2) Data filtering: By applying a modulo operation on the hash values, tuples meeting specific criteria can be filtered. In this example, modulo 11 is employed as the filtering condition, meaning only tuples whose hash values modulo 11 equal 0 will be selected, serving as targets for watermark embedding.

3) Initialization of watermark information: The user-input custom watermark plaintext information is encrypted using the ChaCha20 stream cipher algorithm to generate ciphertext, which is then converted into a binary sequence. The length of this sequence is recorded as the watermark to be embedded.

3.2.2. Watermark Embedding

The watermark embedding phase is the core process of the entire database watermarking technology. Its objective is to embed the user-defined watermark information precisely into the preselected tuples of the database in a manner that is both secure and efficient. When selecting tuples for embedding, the algorithm typically considers the data characteristics and watermark robustness to ensure resistance against various potential attacks, such as data tampering, deletion, or insertion. The detailed flowchart of the watermark embedding process is shown in Figure 3:

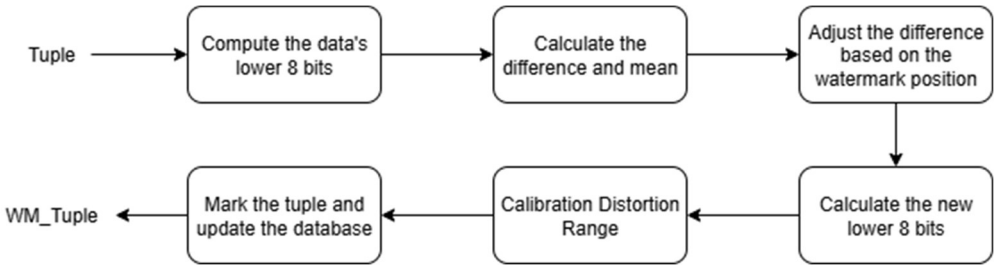

Figure 3. Reversible Database Watermark Embedding Process.

The detailed steps of watermark embedding are as follows:

1) Check tuple ID: Traverse each record in the dataset to verify whether its ID belongs to the tuples designated for watermark embedding.

2) Check the length of the embedded watermark: If the embedded watermark length reaches the preset watermark length, terminate the loop.

3) Calculate the least significant eight bits of the data: First, retrieve the values of columns a, b, and c for the specified tuple from the database. Then, obtain the remainders a8_low, and b8_low by performing modulo 256 operations on a and b, respectively, thereby extracting their least significant eight bits. This step is critical in the watermark embedding process, as it permits modification of the least significant eight bits with minimal impact on the raw data to embed the watermark. Simultaneously, by subtracting these least significant eight bits from the

original values, the high-order parts of a and b are obtained, which provides the necessary separation for the subsequent watermark embedding process.

4) Watermark embedding: First, compute the mean value avg and difference d between a8_low and b8_low, as indicated in formula (3-1). Next, adjust the difference based on the current watermark bit to complete the differential expansion, as indicated in formula (3-2). By utilizing the mean value and the expanded difference, the new least significant eight bits of a and b can be calculated, as shown in formulas (3-3) and (3-4), thereby completing the watermark embedding.

5) Distortion verification range: first confirm that the modified new least significant eight bits of a and b are both within the range of 0 to 255. Only when this condition is met will the least significant eight bits of a and b be merged with their original higher bits to compute the new attribute values. Additionally, a bitwise OR operation is performed on the value of c to mark that the data tuple has been successfully embedded with the watermark.

6) Database update: upon completion of the watermark embedding, the database is updated with the new values of columns a, b, and c to ensure the watermark information is persistently preserved.

4. Analysis of Watermark Performance and Robustness Testing

4.1. Relationship between Watermark Capacity and Degree of Data Distortion

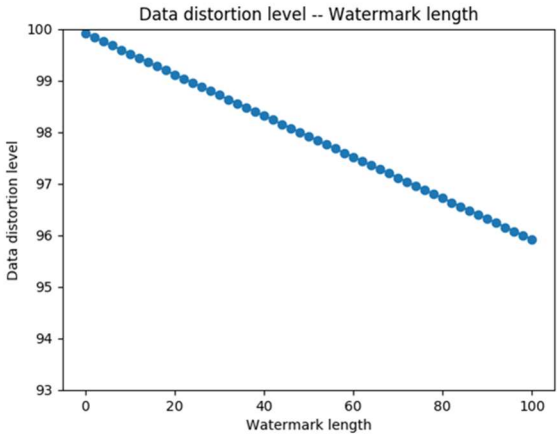

Figure 4. Relationship Between Watermark Capacity and Degree of Data Distortion.

As shown in Figure 4, the relationship between watermark capacity and degree of data distortion is clearly illustrated. The x-axis represents the watermark character length, and the y-axis represents the similarity of the data before and after embedding. It can be observed that as the watermark character length increases, the similarity between data before and after embedding decreases linearly, indicating that embedding more watermark information results in greater modifications to the raw data, thereby causing increased data distortion. To balance watermark capacity and data distortion, it is necessary to identify an optimal point at which the watermark capacity satisfies certain requirements while maintaining data distortion within an acceptable range. This typically requires trade-offs and selection based on the specific application scenario and requirements.

4.2. Tuple Modification Attack

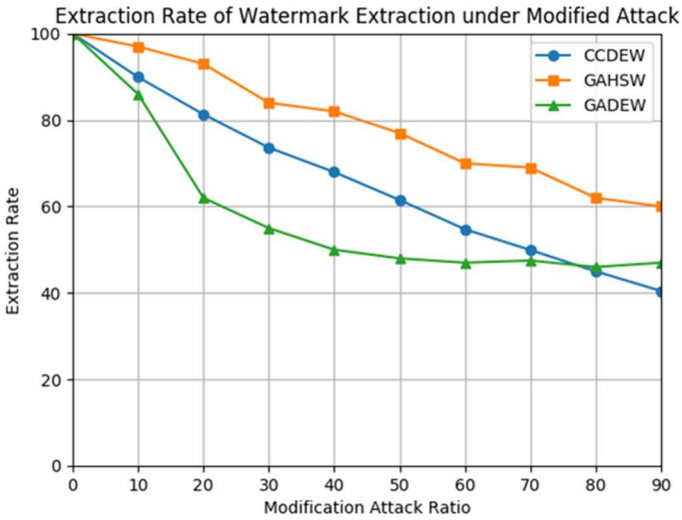

Figure 5. Tuple modification attack experimental results.

Figure 5 shows the experimental results of the tuple modification attack. The experiment simulated modification attack scales from 0% to 90% (x-axis) and calculated the average watermark extraction rate after 100 attacks at each attack scale (y-axis). Experimental results indicate that, as the attack scale increases, the watermark extraction rates of the algorithms all exhibit a decreasing trend, because more tuples containing watermark information are modified. Comparing the proposed CCDEW algorithm with the GADEW algorithm reveals that its decline curve is more gradual, indicating superior robustness against attacks of varying intensities. Even under extreme conditions, with attack scales reaching up to 90%, these algorithms can still maintain approximately a 40% watermark extraction rate, reflecting the effectiveness of the randomized watermark embedding strategy. Effective watermark extraction remains possible despite partial modification attacks on some tuples.

4.3. Tuple Deletion Attack

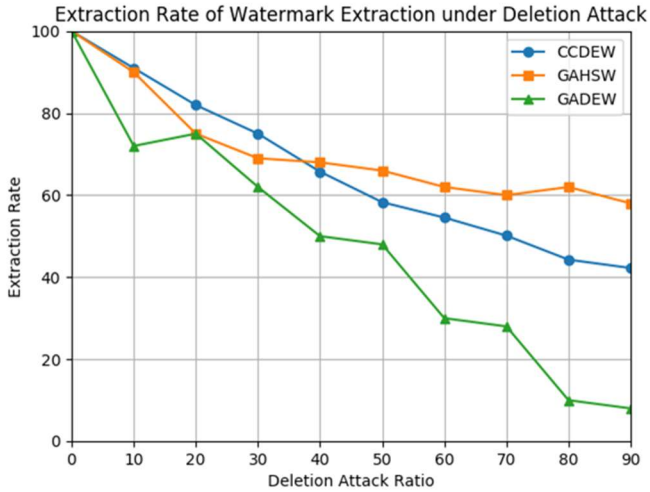

Figure 6. Experimental results of the tuple deletion attack.

Figure 6 presents the experimental results of the tuple deletion attack. The experiment simulated deletion attack scales ranging from 0% to 90% (on the x-axis) and calculated the

mean watermark extraction rate after 100 iterations for each attack scale (represented on the y-axis). By analyzing experimental results, it is possible to clearly observe the performance of the algorithms CCDEW, GAHSW, and GADEW under deletion attacks. The CCDEW algorithm demonstrates relatively stable robustness; even as attack intensity increases, the extraction rate declines gradually, indicating that this algorithm effectively resists deletion attacks and maintains a high watermark extraction rate. The GAHSW algorithm exhibits a rapid decline in extraction rate initially, but the rate of decline slows once the attack ratio reaches a certain threshold. In contrast, the extraction rate of the GADEW algorithm significantly decreases as the deletion attack ratio increases, particularly showing a more pronounced decline at high attack ratios, which suggests weaker robustness against high-intensity deletion attacks. Deletion attacks are considered the most difficult to resist because when the watermark is embedded in attribute values, subset deletion operations inevitably affect all the contained data, resulting in severe watermark loss and consequently a reduction in the watermark extraction rate. The fundamental countermeasure against such attacks is to alter the watermark embedding location, similar to zero-watermarking techniques; however, this solution typically weakens resistance against other types of attacks. The experimental results indicate that all algorithms maintain a watermark extraction rate of 100% when facing tuple addition attacks, demonstrating very high robustness. This is because, during the watermark embedding process, the algorithm determines the precise embedding positions based on the database's primary key values and a specific security key.

5. Conclusion

In the context of rapid advancement in information technology, databases are increasingly exposed to serious data security threats, and traditional database protection methods no longer satisfy current security demands. Therefore, this paper proposes a reversible database watermarking algorithm based on the differential expansion technique, designed to protect database copyright and data integrity while minimizing data distortion and preventing data loss and leakage. The paper also provides a comprehensive understanding and analysis of database watermarking technology, including its fundamental principles, core characteristics, and categories. Through an in-depth study of relational database watermarking technology, the fundamental principles and methods of implementing database copyright protection via database watermarking have been mastered. It proposes a reversible database watermarking algorithm based on the differential expansion technique, which expands the differences between database elements to form new data pairs, thereby embedding watermark information without increasing storage space overhead. The advantage of this method lies in its reversibility, enabling lossless restoration of raw data while extracting the watermark information. Moreover, through careful algorithm design, robustness of the watermark information can be achieved, allowing it to resist malicious attacks and tampering to a certain extent. Experimentally validates the algorithm’s reversibility and robustness, demonstrating its effectiveness in protecting database copyright. Experimental results show that the algorithm can safeguard database security while ensuring data integrity and availability, providing an efficient and reliable technical solution for database security protection.

References

[1] Schyndel R G V, Tirkel A Z, Osborne C F. A Digital Watermark[C]//Image Processing, 1994. Proceedings. ICIP-94. IEEE International Conference.IEEE, 1994. DOI:10.1109/ICIP.1994.413536.

[2] Agrawal R, Kiernan J. Watermarking relational databases. In: Proceedings of 28th VLDB Conference; 2002. p. 155–66.

[3] Di Guandong. Research on Relational Database Watermarking Methods [D]. Qingdao University, 2021. DOI:10.27262/d.cnki.gqdau.2021.002233.

[4] Chai Heyan. Research on Digital Watermarking Technology for Relational Database Copyright Protection [D]. Harbin Institute of Technology, 2020. DOI:10.27061/d.cnki.ghgdu.2020.003473.

[5] Xie Jingsong. Research and Implementation of Reversible Database Watermarking Based on Genetic Simulated Annealing Algorithm [D]. Guilin University of Technology, 2017. DOI:10.27050/d.cnki.gglgc.2017.000268.

[6] Jiang Chuanxian, Li Zhiping, Zhang Tingmao. Reversible Database Watermarking Based on Simulated Annealing Algorithm [J]. Computer Knowledge and Technology, 2017, 13(16):141-143. DOI:10.14004/j.cnki.ckt.2017.1983.

[7] Jiang Chuanxian, Cheng Xiaohui, Xu Xinlei, et al. Reversible Database Watermarking Based on Integer Wavelet Transform [J]. Journal of Guilin University of Technology, 2017, 37(01):191-195.

[8] Duan Jiangbing. Research and Implementation of Reversible Database Watermarking Based on the PSO Algorithm [D]. Guilin University of Technology, 2018. DOI:10.27050/d.cnki.gglgc.2018.000253.

[9] Jiang Chuanxian, Duan Jiangbing, Li Zhiping. Implementation of a Reversible Relational Database Watermarking System Based on the Discrete Particle Swarm Algorithm [J]. Information Technology and Informatization, 2018, (06): 70-72.

[10] Long Xiaoquan. Reversible Database Watermarking Technology Based on SVR Prediction [J]. Journal of Computer Applications and Software, 2017, 34(12): 64-67+137.

[11] Li Yue. Research on Reversible Watermarking for Relational Databases [D]. Tianjin University of Technology, 2023. DOI:10.27360/d.cnki.gtlgy.2023.000516.

[12] Yang Canji. Research on Lossless Database Watermarking Algorithm [D]. Xi'an University of Electronic Science and Technology, 2021. DOI:10.27389/d.cnki.gxadu.2020.002851.

[13] Song Yan, Shen Quanjing, Yang Hongshan. Reversible Database Watermarking Scheme Based on the Cuckoo Algorithm [J]. Computer Applications and Software, 2021, 38(12):10.

[14] Sion R, Atallah M, Prabhakar S. Rights protection for relational data[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 16(12):1509-1525.

[15] Sheh Ab M, Bertino E, Ghafoor A. Watermarking Relational Databases Using Optimization-Based Techniques[J]. IEEE Transactions on Knowledge and Data Engineering, 2007, 20(1):116-129.