**FRAGILE WATERMARKING OF DECISION SYSTEM USING ROUGH SET THEORY**

**Shampa Chakraverty1, Vidhi Khanduja2**

*1Department of Computer Engineering, Netaji Subhas Institute of Technology, India*

[*apmahs.nsit@gmail.com*](mailto:apmahs.nsit@gmail.com)

*2Department of Computer Engineering, Netaji Subhas Institute of Technology, India*

[*vidhikhanduja9@gmail.com*](mailto:vidhikhanduja9@gmail.com)

***Abstract***

***The wheels of modern age are driven by multitudes of databases which serve as repositories of valuable information spread across distributed networks. Artificial Intelligence based applications can fruitfully learn from special repositories called Decision Systems that entail Rough Set Theory to derive high quality classification rules that predict appropriate decisions on freshly gathered data. At the very least, it is necessary to protect the core knowledge that is encapsulated within these Decision Systems from vulnerabilities existing in the Internet. In the past, researchers have dealt at length on robust watermarking for relational databases and made inroads into fragile database watermarking too, but there is virtually no prior work done on watermarked protection of Decision Systems. In this paper, we propose a new fragile watermarking scheme for tamper detection in Decision Systems which detects even the slightest integrity losses that may damage the classification information filtered out from a Decision System. The blind watermarking technique first prepares a secure signature by using the digital encoding of the reducts, the rules and their support values and then embeds this signature into the dataset in a secure manner. We present a theoretical analysis to illustrate the high degree of resilience of our proposed scheme. Experimental results further demonstrate that with even the slightest modification of up to 5% addition, deletion or alteration of tuples, the watermark recreated from the compromised database changes by as much as 54.68% on an average as compared with the originally embedded watermark.***

***Keywords***

***Fragile Watermarking, Decision Systems, Rough Set theory, Tamper Detection, Reducts and Rules.***

1. **INTRODUCTION**

Scientific, technological and social advancements rely heavily upon the systematic compilation of raw data arising from real world events and filtering out intelligent information from them. Sophisticated interfaces are deployed in immersive man-machine environments to gather and send data to central repositories of information, which are subsequently transferred over computer networks and processed. Unfortunately, agents with malicious intent regularly feed upon these very technologies to launch attacks to pilfer or damage data. It is indeed very critical that we protect these databases from any kind of attack, malicious or benign, that may result in compromising their integrity [1].

In order to protect data, database producers embrace cost-effective, self-help Technological Protection Measures (TPMs) which are backed by strong Anti-circumvention laws in many countries [2]. Among security measures, watermarking of digital databases is a widely adopted TPM that provides a reliable means for protecting the ownership of databases [3]-[10], detecting and localizing attempts to tamper with the data [12]-[14], recovering lost data [10] and tracing out database provenance [15].

Decision Systems (DS) are special databases that contain records of objects with condition attributes that characterize them and decision attributes that labels each object with category information. Decision systems in domains such as medicine, finance, meteorology, geography and social networking are regularly utilized as training data for learning rules to make sense out of existing data and forge out classification rules that can be applied to label new objects. Rough Set Theory (RST) is a powerful tool for analyzing a DS and extracting rules that guide in discerning one object from others. It is obvious that in the case of DS, it is this core information regarding discernability and classification of objects that must be protected so that we don’t lose important classification rules due to data tampering.

With the above concern in mind, we develop a fragile watermarking scheme which ensures that even the slightest changes made to a DS that results in concomitant changes in classification rules, are detected immediately. Although several schemes have been reported on watermarking relational databases [3]-[14], to the best of our knowledge, no prior work has been reported on digitally watermarking a DS. Through our work, we seek to fulfil this gap.

The rest of our paper is organized as follows. In section 2, we review prior work carried out in area of digital database watermarking. Section 3 gives theoretical background with a description of rough sets used in our work following by motivation of our work. Section 4 details the implementation of the proposed watermarking technique along with cryptanalysis to illustrate the resilience of the proposed technique. Section 5 presents the experiments performed on the database to check integrity and their results. Section 6 concludes the paper and gives suggestions for future work.

1. **PRIOR WORK**

Literature is rife with interesting works in the domain of robust watermarking for digital databases that provides ownership protection by ensuring that the watermark remains snug within the database, however hard one tries to destroy it [3]-[10]. Robust watermarking techniques strew the watermark bits in profusion all over the database before transmitting them and apply majority voting to recover it after reception. For embedding, researchers have tapped various methods such as using the least significant bits of numeric attributes [3], altering the statistical properties of data [4]-[6] or using non-numeric [7] and categorical attributes [8]. More recent works have proposed distortion-free watermarking [9],[11] and methods to recover important data [10].

Surprisingly, relatively fewer attempts have been made to develop schemes that protect the integrity of relational databases with fragile watermarking. They work upon the principle that the slightest change on data will immediately destroy the watermark, thereby detecting such attempts. Some of the works in this direction include those reported in [12], [13], and [14]. In [12], Yingjiu Li *et.al.* proposed a distortion free fragile watermarking technique for relational databases. The watermark is prepared by using the digital encoding of the entire database to prepare a signature. It is then embedded into the database by reordering tuples rather than by modifying its contents. However, the proposed technique is primary key dependent. Besides, tuple re-ordering is considered to be a malicious modification because the watermark can change irrespective of any change of actual values in the database, which is not desirable. In [13], Guo *et. al.* addressed this problem by proposing a fragile watermarking technique to embed the signature watermark into independent groups of tuples that are decided in a secure manner. However, their technique is also primary key dependent. In [14], Khataeimaragheh *et.al.* proposed a similar technique that can detect and correct distortions by embedding watermarks created from each attribute value, thereby recovering true data. However, the probability of accurately detecting, localizing and rectifying errors reduces drastically when the number of errors exceeds two.

In this paper, we tread a new path by targeting the protection of the classification information that DS databases represent. It is this core knowledge that is encapsulated within a DS that needs to be protected, rather than the raw data. We have applied RST to distil this information in the form of a database of reducts and rules. This database is then utilized to prepare the signature of the information in the original database. The signature now serves as the watermark that is securely embedded as a fragile watermark into the DS, thereby securing its integrity.

1. **BACKGROUND AND MOTIVATION**

In this section, we will briefly look at the essential ingredients of RST to appreciate how it models real world knowledge compiled as raw data in a database. We will also clarify the motivation for our proposal and explain why it is important to protect to protect the classification information of a DS.

* 1. **OVERVIEW OF RST**

RST was introduced by Zdzislaw Pawlak in the early 1980’s [16], [18]. RST is a mathematical framework for analysing data present in the real world. It has been widely used for knowledge discovery in several real life data-centric applications such as medical diagnosis, economics, social media, and recommendation systems. A comprehensive overview of RST and its myriad applications is given in [18], [19].

*Information systems with decision:* An Information System (IS) is represented as, where *U* is non-empty finite set of objects called Universe and *A* is non-empty finite sets of attributes that describe these objects. Any IS that is appended with a decision variable which labels each object with one among a set of known decision classes is called a Decision System (DS). Thus, a DS takes the form, where *d* is the decision attribute and *C* is now the set of conditional attributes. The condition attributes represent observable features and the decision attributes represent the possible outcomes or existing real-world concepts.

RST conducts supervised learning by using the DS as training data to derive sure and possible classification rules that help assign new objects to their appropriate concepts. There are several challenges that must be met in this process. How can we identify the more important attributes that contribute to knowledge significantly and sieve out irrelevant ones? How do we discretize attributes with continuous values so that discernability is preserved? How do we ensure that the derived rules are simple yet powerful enough to classify any new object accurately? What are the objectives for classification? Indeed, these problems as well as their optimal solutions are specific to the target application and must be addressed accordingly.

*Indiscernable objects:* As the first step towards filtering information from the DS, RST evaluates the data in terms of equivalent sets of objects that are indistinguishable from each other in terms of one or more condition attributes. B-equivalence classes are set-members of an Indiscernability relation ℜ(B) defined over a subset of condition attributes *B⊆C*:

ℜ(B) ={(*x, y*) *U*2*| a B, a*(*x*)*=a*(*y*)} (1)

The B-equivalence class of an object *x* is denoted as [x]B.

*Approximations of concepts:* Objects in the Universe can be partitioned into crisp non-overlapping decision classes {K}, each of them representing one real-world concept. The second step in RST is to generate two B-equivalence set approximations for each concept. Whereas the B-lower approximation *B*K *=*{*x*|[*x*]*B* K} is a proper subset of the decision class K, B-upper approximation *X=*{*x|*[x]BK} merely overlaps it [8]. These set-approximations give the basis for formulating sure and approximate classification rules respectively. The ratio |*BX|/* the *Accuracy of Approximation, α.* The Positive Region is the union of the lower approximations of all decision classes. Thus,

Kj (2)

*Reducts:* At first, it may appear attractive to achieve α=100%. This is indeed possible if we select all condition attributes to derive equivalence classes. Deeper inspection reveals that rules derived from such long-spanning set approximations will likely have very few cases supporting them. Hence the *Classification accuracy κ* may drop as new objects get mis-classified. At the other extreme, selecting very few attributes would make the classification very gross with low α, ultimately reducing *κ* due to erroneous classifications. A *reduct* is a minimal set of condition attributes which has the same equivalence class structure as the full set of attributes and therefore preserves the complete discernability information.

The next step in RST is a major challenge, *i.e.* generation of optimal reducts. Given a number of attributes, this is an NP-complete problem that is best tackled by meta-heuristic techniques. The process is guided by application specific objectives and constraints. The primary objective is to maximize the classification accuracy. This may be coupled with subsidiary objectives such as finding minimal reducts, generating a compact set of short rules or finding ensembles of different reducts. Constraints may also be stipulated such as: discern between all objects or achieve a certain level of Information Gain.

*Attribute Reduction:* It is important to eliminate redundant attributes and retain the most important ones for creating reducts. Defining the parameter *γ*(*B,d*) *=* |*POS*(*B,d*)|/|*U*|, the B-significance (*a,d*) of an attribute *a* measures how much of the positive region is sacrificed by its removal from *B*:

(*a,d*) *=* γ(*B,d*) *–* γ(*B–*{*a*}*,d*)/γ(*B,d*) (3)

Redundant attributes with zero or low significance may be removed. A reduct thus combines a sufficient number of important attributes that together model the complete decision-related knowledge encapsulated in a DS.

*Discernability Function:* In order to generate reducts for small data sets, an analytical approach such as Boolean discernibility function may be constructed for each object of the DS. This function is true for all attribute combinations that discern this object from rest of the objects with a different decision. The discernibility function is simplified using Boolean reasoning. Its minimal solutions yields the full set of reducts.

*Reduct optimization:* It is not practically feasible to use analytical approaches to derive the full set of reducts from a large database as they increase exponentially with number of attributes. It is advantageous to use meta-heuristic approaches to derive optimal reducts. We have used Genetic Algorithm (GA) offered by the ROSETTA rough set tool [19] to generate optimal with the set objective of maximizing the positive region.

*Rules:* Reducts are used to synthesize decision rules by overlaying the reducts over the originating decision table and reading off their values. A new object is classified by first identifying those rules whose IF-part (antecedent) matches the features of the new object. Then, existing objects that match the antecedent cast votes for the decision classes in the rules’ corresponding THEN-parts (consequents). The number of votes cast for each rule gives its support value [19].

* 1. **MOTIVATION**

The database may comprise a large number of tuples and attributes, but the relevant information that is conveyed by the data in a DS is the set of reducts and rules that are extracted from it. Indeed, it is the decision relative set of rules derived from a training dataset that distinguishes it as a “decision system”. Reducts may be derived using different objectives, constraints and algorithms. The moot point is that once derived, reducts represent those attributes that significantly retain the discernability between objects in the DS and give a sound basis for generating high quality classification rules.

Therefore, the most important security concern in protecting a DS is to preserve the classification information. If an attacker tampers with the database contents, it may obliterate the information on reducts, the rules or their support values. The watermarking system must ensure that such attempts are made detectable.

In the proposed scheme, we shall generate a signature using an encoding of the extracted reducts and rules and use it as a fragile watermark. The end-objective is to ensure that any change in the database that results in a consequent change in either the reducts or the rules should be immediately detected. In this paper, we take into consideration only sure rules because they have 100% certainty and are practically preferred. However, the watermarking scheme can easily be extended to include possible rules.

1. **TECHNICAL DESCRIPTION**

The proposed watermarking scheme detects any tampering done maliciously or inadvertently to a DS. Table 1 includes various symbols used throughout the ensuing discussion. In this section we will elucidate upon the technical details of our work. Sub-section 4.1 describes the algorithm for fragile watermarking in DS databases and sub-section 4.2 presents a cryptanalysis of its security properties.

* 1. **ALGORITHM FOR FRAGILE WATERMARKING IN DS**

We describe the proposed watermarking scheme in 5 subsections: Extraction of Reducts and Rules, Watermark Creation, Watermark Insertion, Watermark Extraction and Tamper Detection.

Table.1. Table of Symbols

|  |  |
| --- | --- |
| **Symbol** | **Description** |
| *𝒜* | Decision support based dataset |
|  | Watermarked Data set |
|  | Total number of tuples in 𝒜 |
|  | Total number of attributes in 𝒜 |
| *K* | Secret key decided by owner of the dataset |
| *δ* | Fraction of tuples chosen for embedding watermark |
| *ω* | Fraction of attributes selected for embedding watermark |
| *H(t)* | Hash of *tth* tuple |
| *Lw* | Length of the watermark |

* + 1. ***Extraction of Reducts and Rules***

Firstly, the tenets of RST are applied to derive the set of reducts and rules from a given DS, as explained in the previous section. The objectives and constraints of optimizing this process depend upon the target application. In our experiments, we have used the rough set tool ROSETTA [19] to extract reducts and rules from a DS with the aim of maximizing the classification accuracy with full discernability.

* + 1. ***Watermark Creation***

This step creates a unique watermark which serves as a signature of the decision parameters derived from the DS. The process starts by finding the potential locations for embedding the watermark. Algorithm *Select\_Position*(.) shown in Fig. 1 describes this process. It may be noted that the least significant bits *(lsbs)* which are selected for embedding the watermark bits do not participate in watermark creation. *Select\_Position(.)* identifies those embed locations. They are excluded during the next step of watermark creation and utilized again for carrying the watermark bits.

For generality, let us assume that the DS lacks a primary key attribute. Algorithm *Select\_Position*(.) starts by calculating a derived primary key for each tuple (lines 1, 3). To identify each record uniquely, the derived primary key is derived using Eq. 4 below:

|  |
| --- |
| ***Select\_Position*(.)** |
| Input: Dataset *𝒜*  Output: *W\_Status(t), ind(t)*   1. **For** each tuple in 2. Initialize =0 3. Calculate derived primary key (Eq.4) 4. Calculate tuple hash: 5. Assign tuple selection Hash: 6. Assign attribute selection Hash: 7. **If** 8. Select tuple for embedding: Set =1 9. Select embedding attribute index: 10. **End if** 11. **End for** |

Fig. 1. Pseudo-code for selecting the positions to store watermark bits

Where, is most significant bit of the attribute, is the concatenation operator and *Nr* is the number of distinct numeric attributes included in all the reducts extracted.

Any attempt to alter these MSBs by an attacker will result in violation of usability constraints effecting usability of data.

The concatenated MSBs are further concatenated with a secret key *K* which is decided by owner of the DS to yield the tuple hash (line 4). We have applied MD5 algorithm for preparing the tuple hash [20]. Next, two different portions of the tuple hash are extracted for making the following decisions:

* ***Secure selection of watermarking tuples***

The first 20 bits*,* decides whether a given tuple will be used to embed a watermark bit or not (line 5). If the modulus of by *1/δ* yields zero, then the tuple is selected for inserting watermark bit and the tuple’s is set to 1 (lines 7, 8). This process serves two purposes. One, it scatters the watermark throughout the entire DS thus making it difficult for an attacker to locate the watermark bits. Secondly, the use of a secure hash function and secret parameters *K* and *δ* enhances security by concealing the identity of the watermarked tuples from an intruder.

* ***Secure Selection of embedding attributes***

The first 30 bits*,* are utilized for choosing the target attribute for embedding a watermark bit if *t* is selected for this purpose (line 6). The modulus of with is picked up as the target attribute (line 9). This process adds another level of obfuscation by ensuring that the same attribute is not always chosen across different tuples for the purpose of embedding.

|  |
| --- |
| ***Create\_Watermark(.)*** |
| Input: Reducts and Rules of Dataset *𝒜*  Output: Watermark   1. Create reducts signature by using template given in Table 2. 2. Create signature of decision rules by using the template given in Table 3. 3. Calculate Watermark |

Fig. 2. Pseudo-code for watermark creation

After determining the locations for embedding, the watermark is created from the DS by excluding these locations. The *Create\_Watermark*(.) algorithm given in Fig. 2 describes the process of creating the watermark.

* ***Generating Signature of Reducts***

The watermark creation sub-system generates the first part of the signature for capturing the derived reducts (line 1). Table 2 shows the format for part I of the signature derived from all the selected reducts. In any DS, a unique index is assign to each attribute. Let us assume that *N* reducts are selected to form classification rules. We represent as the attribute of the reduct and as the index of .

Given a reduct, its length is recorded in the first column, followed by the indices of the attributes that comprise the reduct. The combined pattern of length and indices for all *N* reducts give the first part of the watermark signature .

* ***Generating Signature of Rules***

Recall that a reduct typically contains multiple equivalence classes. Each equivalence class can map to one or more decision classes, thereby giving sure and/or possible rules to conclude each of these decision classes. Also, there is a unique support value for each rule. Among sure and approximate rules, it is the sure rules that have 100% certainty and are usually preferred. Hence, for the sake of illustration, we assume that only sure rules are selected for creating the rules of database.

Consider a parent reduct *Ri* of length *Li.* A compact representation of all possible rules emanating from the *kth* equivalence class of this reduct has an antecedent and a consequent of the form shown below in Eq. 5:

Where is the attribute of the reduct, is its *xth* value drawn from a set of possible values of , is the decision attribute and and are decision classes.

*Create\_Watermark*(.) next prepares the signature that contain information about these rules (line 2). The above structure is encoded in the signature for rules as illustrated in Table 3. Assume to be the number of equivalence classes in a given reduct *Ri*. All rules derived from a single Equivalence class *Ei1* have a common antecedent which is encoded into the signature by writing their attribute values. *Ei1* can have one, two or as many consequents as the number of decision classes, each of them representing a sure rule with 100% certainty but having different support values. These decision classes and their respective support values are encoded in sequence. Finally this encoding pattern of <antecedent,{decision class, support value}> is repeated for each reduct. The concatenated pattern gives the second part of the watermark signature .

Table.2. Template for preparing the signature of Reducts

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Sig\_A*: Signature for 1..N Reducts** | | | | | | | | |
|  | | | | | **…** |  | **…** |  |
| **Length** | **Index of Attributes** | | | | **…** |  |  |  |
|  |  |  | ... |  | ... | ... | ... | … |

Table.3. Template for generating the signature of rules

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Sig\_B:* Rules for equivalence class of Reduct** | | | | | | | | | | | .. |  |
| **Antecedent** | | | | | **All Consequents** | | | | | |  |  |
| **Decision-class, Support** | | | | | |
|  | ... |  | ... |  | d1 | Sup1 | ... | dn | Supn | ... | ... |  |

In the final step of *Create\_Watermark(.)*, the watermark is prepared by taking a hash of the value of the concatenation of , and the secret key which is known only to the user (line 3). The use of a secret key adds security to the watermark creation process as an attacker cannot determine the watermark without the knowledge of.

***4.1.3* *Watermark Insertion***

Process *Insert\_Watermark*(.)shown in Fig. 3 embeds the watermark into previously securely selected attributes of selected tuples. In order to make a fragile watermark, the entire watermark is embedded only once in the *DS,* such that any change(s) that lead to a change in the classification related decision parameters can be identified. Fig. 3 describes this process.

*Insert\_Watermark(.)* checks the of each tuple. If it is found to be set, the algorithm finds to identify the selected numeric attribute and embeds the next watermark bit into its *lsb* (line 2-4). A single bit is embedded in each tuple and the process stops when all watermark bits have been embedded.

Once the DS is armed with its signature watermark *W*, it is free to travel through the internet where it may be attacked by malicious agents or noise. Under these circumstances, its integrity may be compromised with before it reaches its final destination. Let us denote such a database as

* + 1. ***Watermark Extraction***

The watermark extraction process outlined in the pseudo-code *Extract\_*Watermark(.) in Fig. 4 is responsible for reliably extracting the embedded watermark from a suspected watermarked database. The proposed watermark extraction technique is blind as it does not require the original database for extracting the watermark from the suspected database.

|  |
| --- |
| ***Insert\_Watermark*(.)** |
| Input: Dataset *𝒜* , Watermark *W*  Output: Watermarked dataset   1. Initialize *count* =0 2. **For** each tuple in repeat: 3. **If** 4. Embed  the watermark into *lsb* of selected attribute 5. Increment *count* by 1. 6. **If** 7. **Go to line 10** 8. **End if** 9. **End if** 10. **End for** |

Fig. 3. Pseudo-code for embedding watermark into *DS*

|  |
| --- |
| ***Extract\_Watermark(.)*** |
| Input: Suspected watermarked database.  Output: Extracted watermark W’   1. Call *Select\_Position(.)* 2. Initialize *count* =1 3. **For** each tuple in repeat the following steps 4. **If** 5. Extract the watermark bit from the *lsb* of selected attribute 6. Increment *count* by 1. 7. **If** 8. **Go to line 10** 9. **End if** 10. **End if** 11. **End for** |

Fig. 4. Pseudo-code for extracting the watermark

The subroutine *Extract\_*Watermark(.) starts by calling *Select\_Position*(.) to re-calculate the original embed locations (line 1). It then extracts the watermark bit from the *lsb* of each earmarked attribute of selected tuples (line 4, 5). Variable *count* keeps the tracks of number of bits extracted. As soon as watermark bits are extracted, the process terminates.

* + 1. ***Tamper Detection***

Tamper detection takes place at the receiver side. Process *Extract\_Watermark*(.)is invoked to generate the extracted watermark be *WX.* Next,process *Create\_Watermark*(.) is invoked to re-generate the watermark of the suspected database using its contents. Let the re-created watermark be denoted as *WR*. Let us consider the following cases:

1. There were no integrity attacks. If neither the data nor the watermark were changed, then *WR=W* and *WX=W.* Thus, the two watermarks will match (*WR=WX*).
2. The content of the DS was changed but not the embedded watermark. If changes to the DS resulted in some change to either the reducts or the rules, then the re-generated watermark will not be the same as the original watermark, *i.e.* *WR≠W.* Thus, *WX ≠ WR,* and the tampering event will surely be detected.
3. The raw data in the DS was not changed but the embedded watermark was changed. In this case, the reducts and rules will remain the same. Hence *WR=W* but due to tampering, *WX* changes. Hence, *WX ≠ WR ,* and the tampering is detected.
4. The content of the DS was changed, but that did not result in a corresponding change in the decision rules or reducts. Hence, *WR = W=WR.* Since the desired information encapsulated within the DS was not affected despite tampering, this result is acceptable.
5. Both reducts/rules as well as the embedded watermark were changed. This case has a very remote chance of the two new watermarks produced as a result of the changes to the DS and the embedded watermark respectively, turns out to be exactly the same. Hence *WX ≠ WR* and the tampering is detected.

From the above, it is clear that the watermark is highly fragile. Any changes made to the dataset that affects the reducts and/or rules or the embedded watermark or both can be immediately detected.

* 1. **CRYPTANALYSIS**

Let us now assess various scenarios when an attacker Mallory or a cryptanalyst first tampers either the database or the embedded watermark or both and then manages to cover it up by cleverly making matching changes in the complimentary part so as to escape detection. In her attempt to dodge, an attacker will try her best to engineer changes so that the recreated and extracted watermarks become equal. We now perform a cryptanalysis of different attack scenarios. As usual, we assume the algorithm for embedding the watermark is publicly known but the owner’s secret parameters are kept hidden.

* ***Scenario 1:*** Let an attacker alter only the DS and not the embedded watermark. If such tampering changes the reducts or the rules or their support values, then it will result in a new recreated watermark *WR′*. The attacker will now try to make changes to the embedded watermark so that when it is extracted, *WX′* becomes equal to WR′. We now show that the probability of the attacker being able to do so is negligible.

Note that in this scenario, the attacker’s main challenge is to reach the correct positions where the watermark bits are placed. Since tuples and attributes are selected by using a cryptographic hash function, the output values are completely randomized with uniform distribution. In the absence of any knowledge of the secret parameters *K,δ* and *ω*, an attacker has no means to guess whether a tuple is selected or not by simply looking into the DS. She can only pick up the tuples randomly from the total number of tuples in the hope that would turn out to be correct and then make guesses on the embedding attributes within these tuples.

As watermark of length, is embedded once in selected tuples, the total number of embedded tuples is. Therefore, the probability that the first tuple chosen by the attacker is correct (*i.e.* it was used for embedding) is:

The single attribute within this tuple that carries a watermark bit, was also selected by using a cryptographic hash function and secret parameter ω. Hence it is completely randomised and the attacker has no option but to make a guess. Since exactly one attribute is selected from among, the probability of finding it correctly is:

The composite probability of finding the first selected position is equal to: . Next, the probability of finding the second selected tuple correctlyis The composite probability of finding the second position is therefore .

Thus, the overall probability of finding all the embedded positions within the database correctly, is:

Assuming, Nt=600, Lw=128, Na=5 we get Pep=1/(4.64\*e+133\*2.94\*e+89) which is virtually nil. Hence, an attacker will simply not be able to detect all embedded positions correctly.

Suppose the changes made to DS did not have any impact on the reducts and rules generated, or their support values. If the classification information is not affected in any way, the recreated watermark will not change and the tamper attempts will go completely unnoticed. However, note this does not violate our original objective of protecting the classification knowledge. The changes resulting in alteration to insignificant portions of the data and may be ignored.

* ***Scenario 2:*** The attacker has tampered the embedded watermark so that the extracted watermark now changes to *WX′* and will now tries to work backwards to alter the DS so that the recreated watermark *WR′* becomes the same as the watermark extracted *WX′* after tampering it.

This kind of attack is known as preimage attack [20]. Taking hash along with the concatenation of key makes the watermark creation procedure secure. The key is kept secret and hash is a secure one way function. Moreover, it possesses a strong Avalanche effect. Even with a single bit change in input, large number of bits changes in the hashed output. Hence, it is difficult to guess the input given the output of the secure hash functions.

Now, the only option left is a brute force attack. An attacker will try out all possible values of the *hash(x)* such that *hash(x)= WX′.* To find all possible values of this combination is not feasible in real time. The estimated probability is

Where, *Lw* is length of output of the hash function in bits. Taking MD5 hash algorithm, we get a 128 bit hash. Thus, Pe=2.9387e-39 which is negligible.

* ***Scenario* 3:** The attacker has randomly changed both DS as well as the watermark. Now she does not have any option left to match them up but, the extracted and the embedded watermarks (both now changed) come out to be the same by sheer chance. The probability that the signature of the new DS turns out to be the same as new watermark by sheer chance is the same as probability that bits of both watermark matches. Since each bit can be equal to 0 or 1 and is independent of other bits, its probability is 1/2. Thus, probability that all bits matches is which is indeed negligible.

1. **EXPERIMENTAL RESULTS**

We performed our experiments on Intel Core™ i7 2.30 GHz with 64-bit operating system. We implemented watermark insertion and extraction subroutines in MATLAB 7.8.0. The dataset contains the preprocessed and filtered sessionized data for the main DePaul CTI web server [21]. The data is based on a random sample of users visiting this site for a two week period during April2002. The database contains 5 attributes and 20509 tuples. This dataset is processed further to obtain decision system on which experiments were performed.

We used Rosetta tool version 1.4.40 to implement RST. This freeware software helps us analyze rough sets and their behavior. In our experiments we used Genetic Algorithm for reduction of the dataset and full indiscernibility for the generation of rules. We analysed the integrity of the watermarked against various attacks.

* 1. **INTEGRITY ANALYSIS**

Consider an attacker Mallory who tries to alter the in an undetectable manner. Mallory can either delete the tuple or add new tuples or alter the values of the watermarked inorder to tamper the. We have performed the experiment against all these possibilities and categorized them accordingly as:

* + 1. ***Tuple Addition Attack:***

In this attack, Mallory tries to alter the by adding some spurious tuples in. In this way, she hopes the watermark embedded in the remains untouched. In our proposed technique, the watermark creation procedure is used to check the integrity of a suspected. It creates a signature of the; hence any modification to theis bound to alter its signature. Therefore, if Mallory tries to make any changes to, then on running the watermark creation procedure we will always obtain a different watermark from the embedded one. The graph in Fig.5 shows on adding only 5% of total tuples, there is as much as 54.68% change in the recreated watermark.

* + 1. ***Tuple Alteration Attack:***

Mallory tries to modify the by randomly altering some data bits such that the data is not rendered completely useless. To simulate this kind of attack, we altered the and recreated the watermark using C*reate\_Watermark(.)* procedure. The graph in Fig.6 shows significant changes in the recreated watermark as a result of such alterations. There is 60.9% change in recreated watermark on altering 5% of tuples. Thus our technique is resilience to this attack.

|  |  |
| --- | --- |
| Fig.5. Simulation results of Tuple Addition attack showing percentage of matched watermark with new added tuples. | |
| Fig.6. Simulation results of Tuple Alteration attack showing percentage of matched watermark with altered tuples.   |  | | --- | | Fig.7. Simulation results of Tuple Deletion attack showing percentage of matched watermark with deleted tuples. |  * + 1. ***Tuple Deletion Attack:***   Mallory tries to delete tuples from thewith an intention of altering the without affecting the watermark. The graphical result shown in Fig. 7 reveals that even if 5% tuples are deleted, the recreated watermark changes significantly upto 53.9%. Thus, one can easily claim that the was tampered with.   1. **Conclusion**   In this paper, we proposed a novel fragile watermarking technique which detects integrity losses in the knowledge that is implicit in a DS, regarding the ability to discern between objects. This is achieved by preparing a signature of this core information securely, using reducts and rules derived from the using RST. The watermark is then embedded in securely selected positions within the . The proposed technique is blind as it does not require the original *DS* during watermark extraction. We performed experiments to simulate tuple addition, deletion and alteration attacks. The results clearly indicate that even with the slightest modification of the , the watermark re-created from it is changed magnificently. Specifically, a meagre 5% alteration in the brings about 60.9% change in recreated watermark. Similarly, on 5% addition of tuples, the watermark changes by 54.68% and on 5% deletion of tuples, the watermark changes to 53.9%. Hence, we can confidently claim that the was tampered with.  Our future work will focus on localization of the tamper occurrences. We are also working on regeneration of pivotal attributes after damages are detected. | |

**ACKNOWLEDGEMENTS**

The authors wish to thank Pramit Mallick and Saloni Kapoor for their collaborative efforts towards the work present in this paper. Both of them are studying at Netaji Subhas Institute of Technology, Delhi, India under department of Computer Engineering.

**REFERENCES**

1. Elisa Bertino and Ravi Sandhu, "Database Security—Concepts, Approaches, and Challenges", *IEEE Trans. on Dependable and Secure Computing*, Vol.2, No.1, pp.2-19, 2005.
2. Ian Brown E, “The evolution of anti-circumvention law”, *International Review of Law, Computers and Technology*, Vol.20, Issue 3, pp. 239-260, 2006.
3. Rakesh Agrawal, Peter J. Haas, J. Kiernan, “Watermarking relational data: framework, algorithms and analysis,” *VLDB J.* Vol. 12(2), pp.157-169, 2003.
4. R. Sion, M. Atallah , S. Prabhakar, “Rights protection for relational data,” *IEEE Trans. on Knowledge and Data Engineering*, Vol.16, No.12, pp.1509-1525, 2004
5. Kamran, M., and Farooq, M., “An information Preserving Watermarking Scheme for right protection of EMR systems”, *IEEE Trans. on Knowledge and Data Engineering*, Vol. 24, No.11, pp. 1950–1962, 2012.
6. V.Khanduja, O.P.Verma and S.Chakraverty, “Watermarking Relational databases using Bacterial Foraging Algorithm”, *Multimedia tools and Applications, Springer*, Online, October 2013, DOI: 10.1007/s11042-013-1700-9.
7. R. Bedi, A. Thengade, V.M. Wadhai, “A New Watermarking Approach for Non-numeric Relational Database,” *IJCA*, Vol. 13, pp.37-40, 2011.
8. R. Sion, M. Atallah, S. Prabhakar, “Rights protection of categorical data,” *IEEE Trans. on Knowledge and Data Engineering*, vol.17,no. 7, pp. 912-926, 2005.
9. Vidhi Khanduja,Shampa Chakraverty, OmPrakash Verma, “Robust Watermarking For Categorical Data,” *in Proc. IEEE International conf. on Control, Computing, communication and materials*, India, pp. 174-176, 2013.
10. V.Khanduja, S.Chakraverty, O.P.Verma, Rakshita and S.Goel, “A Robust Multiple Watermarking Technique for Information Recovery”, *in Proc. IEEE International Conference on Advanced Computing*, India, pp.250-255, 2014.
11. S. Bhattacharya, A. Cortesi, “Database authentication by distortion free watermarking,” *in Proc. International conf. on Software and Data Technology*, pp.219-226, 2010.
12. Y.Li, H.Guo and S.Jajodia, “Tamper Detection and Localization for Categorical Data Using Fragile Watermarks,” *in Proc. of the 4th ACM Workshop on Digital Rights Management,* Washington DC, USA, pp. 73-82, 2004.
13. H.Guo, Y.Li, A.Liu and S.Jajodia, “A fragile watermarking scheme for detecting malicious modifications of database relations”, *Information Sciences, Elsevier*, Vol. 176, Issue 10, pp.1350–1378, 2006.
14. H. Khataeimaragheh and H. Rashidi, “A Novel Watermarking Scheme For Detecting And Recovering Distortions In Database Tables”, *International Journal of Database Management Systems*, Vol.2, No. 3, pp.1-11, 2010.
15. Martin Schäler, Sandro Schulze, Ronny Merkel, Gunter Saake, and Jana Dittmann,  "Reliable Provenance Information for Multimedia Data Using Invertible Fragile Watermarks ",*in Proc. BNCOD 2011, LNCS 7051*, pp. 3–17, 2011.
16. Z.Pawlak, “Rough Sets” *International Journal of Computer and Information Sciences,* Vol. 11, Issue 5, pp.341-356, 1982.
17. Komorowski, L.Polkowski and A.Skowron, “Rough Sets: A Tutorial”, *Rough Fuzzy Hybridization- A new Trend in Decision Making, Springer-Verlag*, pp.3-98, 1998.
18. Z.Pawlak, “*Rough Sets: Theoretical Aspects of Reasoning about Data, System Theory, Knowledge Engineering and Problem Solving*”, Vol. 9, Kluwer Academic Publishers, Dordrecht, 1991.
19. Torgeir R. Hvidsten , “A tutorial-based guide to the ROSETTA system: A Rough Set Toolkit for Analysis of Data”, 2nd Ed, 2010
20. William Stallings, “*Cryptography and Network Security”,* fourth ed., Pearson Education India, Chapter-2, pp. 40-42, 2005.
21. Online dataset Available at http://facweb.cs.depaul.edu/mobasher/classes/ect584/resource.html.