Fragile Watermarking of Decision System Using Rough Set Theory

Article · March 2018

DOI: 10.1007/s13369-018-3120-7

CITATIONS
4

READS
71

2 authors, including:

Vidhi Khanduja
Netaji Subhas Institute of Technology
14 PUBLICATIONS **138** CITATIONS

SEE PROFILE

FRAGILE WATERMARKING OF DECISION SYSTEM USING ROUGH SET THEORY

Vidhi Khanduja Shampa Chakraverty
Department of Computer Engineering
Netaji Subhas Institute of Technology, Delhi

Abstract—The wheels of modern society are driven by multitudes of databases that serve as repositories of valuable information. Several applications can fruitfully learn from special repositories called Decision Systems.DS are databases that contain records of objects described by condition attributes and labeled by decision attributes that categorize them into distinct classes. Rough Set Theory can be applied to derive high quality classification rules from them to predict accurate decisions on freshly gathered data. For security, it is necessary to protect this core information from vulnerabilities in the Internet. No prior work is done on watermarked protection of Decision Systems. To fulfill this gap, we propose a new fragile and blind watermarking scheme for tamper detection in Decision Systems which detects even the slightest integrity losses that may damage the classificatory information encapsulated within a Decision System. The technique characterizes the type of attack and then localizes the perturbation upto an attribute's value level. In case of alteration in reducts, proposed technique can recover the original value. The watermarking technique first prepares secure signature by encoding the information on reducts, rules and their support values. It securely embeds this into dataset. We present theoretical analysis to illustrate the high degree of fragility of our proposed scheme to different kinds of attacks. Experimental results demonstrate that with the modification of up to 50% addition, deletion or alteration of tuples, the watermark recreated from the compromised database reflects the changes close to 50% on an average, thus facilitating immediate detection.

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

1. INTRODUCTION

In modern society all scientific, technological and social advancements rely heavily upon the systematic compilation of data and the information filtered from them. Innumerable interfaces are deployed in immersive man-machine environments to gather and transmit data to central repositories where they are stored, processed and are subsequently transferred over computer networks. However, agents with malicious intent regularly feed upon these very technologies to launch attacks to pilfer or damage the data. It is necessary to protect these databases from any kind of malicious or benign attack that may result in compromising their integrity [1].

In order to protect data, database producers embrace cost-effective, self-help Technological Protection Measures (TPMs). These 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 various security goals such as ownership protection [3-8], tamper detection [9-10], information recovery [11-12] and

database provenance [13]. Other security techniques such as encryption and hashing encode the data by making it meaningless. They intend to serve the purpose of confidentiality or secretly sending the data/message over the channel. However, digital watermarking works on the principle of securely embedding the marks within the data/database such that usability of data remains unaffected. Additionally, an attacker or observer cannot depict that a watermark is actually present in the database. The watermark remains within the database inseparably providing a proof of the ownership or a signature to detect tampering for integrity of database or to track the people who may have obtained the content legally and are illegally redistributing it [3].

Like any other database, Decision Systems (DS) are transferred through the internet and are therefore, subject to security threats that may result in compromising their integrity. An Information System that is appended with a decision variable that labels each object with one among a set of known decision classes is called a DS. Thus, a DS takes the form $\mathcal{A} = (U, C \cup \{d\})$, where U is non-empty finite set of objects called Universe, C is the set of conditional attributes and d is the decision attribute such that $d \notin C$. The condition attributes represent observable features and the decision attributes represent the possible outcomes, *i.e.* distinct real-world concepts.DS in diverse domains such as medicine, finance, meteorology, geography and social networking are regularly utilized as training data by machine learning algorithms.

It is worth noting that in terms of security of a DS, it is more important to protect the crucial classification information that is encapsulated within a DS rather its raw data. This would ensure that we don't lose classification rules due to unwarranted data tampering. With this concern in mind, we propose a novel scheme for fragile watermarking of DS. It detects even the slightest changes occurred in a DS that results in concomitant changes in classification rules derived from it. Although several schemes have been reported on watermarking relational databases, to the best of our knowledge, no prior work has been reported on digitally watermarking a DS. Through our work, we seek to fulfill this gap.

The rest of the paper is organized as follows. In section 2, we review prior work carried out in the area of digital database watermarking. Section 3 gives the theoretical background for our work. In Section 4, we explain the architecture of the proposed watermarking technique, followed by a security analysis to illustrate the fragility of the proposed technique in Section 5. Section 6 presents experimental results, and we conclude the paper in Section 7.

2. LITERATURE SURVEY

Literature is rife with interesting works in the domain of robust watermarking for digital databases. They provide ownership protection by ensuring that the watermark remains intact within the database however hard one tries to destroy it [3-8, 11-12]. Robust watermarking techniques store the watermark bits in profusion all over the database before transmitting them. After reception, a majority voting is applied to recover the watermark. Methods proposed for robust watermarking includes altering the least significant bits of the attributes [3] and altering the statistical properties of data [4, 6]. More recent works have proposed distortion-free watermarking [14-15] and additional methods to recover important data [11-12].

Relatively fewer work exist on fragile watermarking to protect the integrity of databases. They work upon the principle that the slightest change made to the database will immediately destroy the watermark. Some of the works in this direction include those reported in [9-10]. In [9], the authors propose a distortion free fragile watermarking technique for relational databases. The watermark is prepared by using the digital encoding of the entire database as its signature which is embedded into the database by reordering tuples rather than by modifying its contents. However, tuple reordering is considered to be a malicious modification because the watermark can change irrespective of any change of actual values in the database. In [10], 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 primary key dependent.

In this paper, we tread a new path by protecting the classificatory information that is hidden in a DS. For classification purposes, it is this core knowledge that needs to be protected, rather than raw data. Thus, we propose a technique that detects, characterizes, localizes and recovers the true information from the tampered database.

3. THEORETICAL BACKGROUND

Our work is founded upon the principles of Rough Set Theory (RST).RSTis a powerful tool for conducting an approximate analysis of DS and extracting classification rules that guide in discerning objects. These rules can be applied to label new objects. In this section, we will overview the tenets of RST. Introduced by Zdzislaw Pawlak in the early 1980's [16-17], RST gives a mathematical approach for analyzing real-world data within an approximate

framework. 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].We give the introduction of reducts and rules generated from DS using RST.

3.1. Information Systems with Decision

As discussed earlier, an Information System (IS) is represented as $\mathcal{A} = (U, A)$, where U is non-empty finite set of objects called Universe and A is non-empty finite sets of attributes that describe these objects. An 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 $\mathcal{A} = (U, C \cup \{d\})$, where d is the decision attribute such that $d \notin C$ and C is the set of conditional attributes. The condition attributes represent observable features and the decision attributes represent the possible outcomes, *i.e.* distinct real-world concepts [19].

3.2. Equivalence Sets

In order to reduce the number of cases in a DS and represent groups of similar objects with a template, equivalence classes are generated. A B-equivalence set contains all those objects that are indistinguishable from one another other in terms of a subset B of condition attributes. The Indiscernability relation I(B) defined over a subset of condition attributes $B \subseteq C$ is given by:

$$\mathrm{I(B)} = \{(x, y) \in U^2 \mid \forall a \in B,\ a(x) = a(y)\} \qquad (1)$$

The B-equivalence class of a specific object x is denoted as $[x]_B$.

Approximations of Concepts: The decision attribute represents real-world concepts. Objects in the Universe can be partitioned into crisp non-overlapping decision classes $\{X\}$, each of them representing a distinct concept. As per the basic principles of RST, each concept is perceived in terms of two set approximations. The B-lower approximation $\underline{B}(X)$of concept X is collection of all those B-equivalence classes which are proper subsets of X:

$$\underline{B}(X) = \{x \mid [x]_B \subseteq X\} \qquad (2)$$

Thus, the lower approximation $\underline{B}(X)$ *surely* belongs to the concept X. A B-upper approximation is a collection of all equivalence classes that overlap with X:

$$\overline{B}(X) = \{x|[x]_B \cap X\} = \phi \quad (3)$$

Thus the upper approximation $\overline{B}(X)$*possibly* belongs to the concept X. The *Accuracy of Approximation,* $\alpha$is given by the ratio:

$$\alpha = \underline{B}(X)/|\overline{B}X)| \quad (4)$$

The Positive Region is the union of the lower approximations of all decision classes. Thus,

$$POS(B, d) = \cup_j \underline{B}(X_j) \quad (5)$$

3.3. Reducts

A *reduct* is a minimal set of condition attributes, which gives the same equivalence class structure as the full set of attributes and therefore, preserves the essential discernability information. The major challenge in RST is to generate a set of optimal reducts. This is an NP-hard problem [18]. Hence, it is best tackled by meta-heuristic techniques such as Genetic Algorithm (GA) or Johnsons algorithm [20-21]. This process is guided by application-specific objectives such as maximizing the accuracy of approximation, maximizing the classification accuracy, generating a compact set of short rules and deriving ensembles of reducts.

For creating reducts, it is important to eliminate redundant attributes and retain the most important ones. The γ-parameter indicates the proportion of all objects that fall in a positive region, given a reduct B:

$$\gamma(B, d) = \frac{|POS(B,d)|}{|U|} \quad (6)$$

The B-significance $\sigma_B(a,d)$ of an attribute *a* measures how much of the positive region is sacrificed by its removal from *B*:

$$\sigma_B(a,d) = \gamma(B,d) - \gamma(B-\{a\},d)/\gamma(B,d) \quad (7)$$

Redundant attributes with zero or low significance may be removed.

3.4. Rules

The lower and upper set-approximations derived from reducts yield sure and approximate classification rules respectively [16, 19]. A decision rule is induced by an object x from the reduct *B*:

$$RULE: b_1(x), b_2(x),\ldots, b_n(x) \rightarrow d(x) \quad (8)$$

It has an antecedent comprising the sequence of condition-attribute values: $b_1(x)$, $b_2(x),\ldots, b_n(x)$ where $b_i \in B$. It implies the consequent $d(x)$. Let $[x]_d$ be the set of objects, which have the same value of the decision attribute as that of x. Thus, the support $sup(B,d)$ is given by:

$$\sup(B,d) = |[x]_B \cap [x]_d| \quad (9)$$

A new object is classified by first identifying those rules whose antecedent matches the features of the new object. Existing objects that match the antecedent cast votes for the decision classes implied by the rules' own consequents. The number of votes cast for each rule gives its support value. The new object is assigned the winning decision class.

4. ARCHITECTURE OF THE PROPOSED TECHNIQUE

We propose a fragile watermarking scheme that detects any malicious or inadvertent tampering to a DS damaging its underlying classificatory information. Figure 1 shows the flow of the watermarking scheme. The proposed watermarking scheme comprises of four phases; Watermark Preparation, Watermark Insertion, Watermark Extraction and Decision Generation [22]. Table 1 includes various symbols used throughout the ensuing discussion.

Information is framed in the form of a decision system $\mathcal{A}$. We apply RST to extract the reducts, the rules and their support values. Watermark Preparation phase utilizes these features to prepare a signature of a DS using secret key $\mathcal{K}_s$. The signature now serves as a watermark $\mathcal{W}_s$ that is securely embedded as a fragile watermark into a DS in Watermark Insertion phase. The secret parameters involved in selection of embed positions are tuple selection parameter γ, number of candidate attributes β and secret key $\mathcal{K}_s$. The proposed scheme secures informational integrity. Decision Generation phase compares a watermark $\mathcal{W}_g$ that is regenerated from the rules and reducts of the suspected decision system with the extracted watermark $\mathcal{W}_x$. If there is a mismatch, this phase takes the decision that database is

tampered and localizes the mismatch. We now elucidate the technical details of the proposed watermarking scheme.

Figure 1. Block diagram of a proposed fragile watermarking model for Decision System

Table.1. Table of Symbols

4.1.Watermark Preparation

This step creates a unique watermark which serves as the signature of decision parameters derived from $\mathcal{A}$. In order to make a fragile watermark, the watermark is prepared from $\mathcal{A}$, such that any alteration(s) that leads to a change in the classification-related decision parameters can be identified. Figure 2illustrate the pseudo-code of *Create_Watermark*(.) that generates the watermark from $\mathcal{A}$. We first extract the reducts and rules and then, using these prepare the watermark $\mathcal{W}_s$ (line 4).The template of the watermark is divided into three portions shown in figure 3. The first part of a watermark contains codified information on the reducts that are generated. The second part encodes information on the rules that are derived. A secret key $\mathcal{K}_s$ is concatenated with other two portions to create a secure watermark. We now discuss each of the steps involved in *Create_Watermark(.)* subsequently.

Figure 2. Pseudo-code for creating a watermark $\mathcal{W}_s$ from dataset $\mathcal{A}$

A. Extract Reducts and Rules: As explained in section 3.3 above, *reduct* is a minimal set of condition attributes that preserves the essential discernability information. For creating reducts, we eliminate redundant attributes and retain the most important ones.

Using these reducts, decision rules are generated. Multiple equivalence classes can be extracted from the attributes in reducts. Each equivalence class can map to one or more than one decision classes, thereby giving rules.

Figure 3. Template of watermark $\mathcal{W}_s$

B. Generate Signature of Reducts: We prepare the first component of the signature by encoding the properties of all the reducts derived from the $\mathcal{A}$ (line 1-2 in fig.2). Figure4(a)shows the format for the first part Sig_A.

Figure 4. Template for generating Sig_A and Sig_B

Each attribute in $\mathcal{A}$ has a unique 1-dimensional index. Recall that a reduct is a collection of significant attributes. Let us assume that *N* reducts are selected after the reduct optimization process. We represent $A_{i,k}$as the k^{th} attribute of the i^{th} reduct and L_i as number of attributes in reduct R_i. The function $I(A_{i,k})$ converts the 2-dimensional index$A_{i,k}$ to its single 1-dimensional index in $\mathcal{A}$. Let us take the first reduct R_1. Its length L_1 is recorded in the first column. This is followed by the indices of the attributes that comprise the reduct. This pattern is repeated for each and every reduct in sequence. The combined pattern of lengths and the attribute indices for all *N* reducts produce the first part of the watermark signatureSig_A.

C. Generate Signature of Rules: *Create_Watermark*(.) next prepares the signature that contains information about the rules generated in line 3 of fig.2. Figure4(b)shows the format for the second part Sig_B.

Consider a reduct R_i of length L_i. A compact representation of any rule emanating from the equivalence class of this reduct has an antecedent and a consequent of the form shown below in equation (10).

$$\boldsymbol{IF}(A_{i,1} = V_x) \dots AND \dots (A_{i,k} = V_y) \dots (A_{i,L_i} = V_z)\ \boldsymbol{THEN} d = d_1(Sup_1) OR d = d_2(Sup_2) \dots \quad (10)$$

Where $A_{i,k}$is the k^{th}attribute of the i^{th}reduct, V_y is drawn from a set of possible values of$A_{i,k}$, $L_i$is the length of reduct R_i, d is the decision attribute and d_1 and d_2 are decision classes. Sup_1 represents support of the d_1 and Sup_2 represents support of the d_2. The above structure is encoded in the signature of rules as illustrated in figure 4(b).

Assume N_q to be the number of equivalence classes in a given reduct R_i. All rules derived from a single equivalence class E_{i1}has common attributes in the antecedent which is encoded into the signature by writing their attribute values. Let, N_{Ru} be number of rules having common attributes in antecedent w.r.t. a particular reduct. E_{i1} 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

< N_{Ru}, antecedent,{decision class, support value}> is repeated for each reduct. The concatenated pattern gives the second part of the watermark signature Sig_B.

We have used the rough set tool ROSETTA [21] to extract reducts and rules from $\mathcal{A}$with the aim of maximizing the classification accuracy while maintaining full discernability and recorded the results in Table 2. The details of datasets used are discussed in Section 5.From the table we infer following points:

1. Number of rules obtained is extremely large. Average ratio of rules with the number of tuples is 7:8.
2. The watermark formed using these rules contains large number of bits. We cannot embed all the rules as it is into DS because of its large length. Thus, we need to compress them for the purpose of embedding the watermark robustly.
3. On an average, number of reducts is nearly half of the number of attributes. This means nearly half of the non-reduct attributes can be targeted to embed watermark. Thus, we have enough potential locations to conceal watermark redundantly.

Table 2: Obtaining reducts and rules from DS.

D. Generate Watermark: We generate the watermark $\mathbb{W}_s$ that acts as a signature of $\mathcal{A}$ by concatenating Sig_A, Sig_B and the secret key$\mathcal{K}_s$ and then compressing this concatenated value (line 4). The use of a secret key $\mathcal{K}_s$ adds security to watermark creation process because an attacker cannot determine the watermark without the knowledge of$\mathcal{K}_s$.In the proposed work, we first compress the signatures to obtain final watermark $\mathbb{W}_s$. Experiment section 5.2 shows the result of compressing $\mathbb{W}_s$ using different algorithms. We implemented PPMd algorithm to compress the signatures so as to get the minimum possible watermark bits. PPMd is an adaptive statistical data compression technique suitable for text based content. Predicion by partial matching models applies a set of previous symbols in the uncompressed symbol stream in order to predict the next symbol in the text stream [27-28].

4.2. Watermark Insertion

This process embeds the watermark into securely selected attributes of earmarked tuples. We select the attributes that does not belong to reducts as candidate attributes for concealing the watermark. We now explain each of the sub-processes involved.

A. SelectEmbed Positions: The process of embedding a watermark starts with identification of suitable locations for embedding the watermark. Figure 5 represents the pseudo-code *Select_Positions*(.) that selects the positions securely. The attributes that are excluded from reducts acts as candidate attributes for embedding. The scheme selects γ fraction among N_t tuples and two attributes out of β numeric attributes as candidates that acts as carriers of watermark bits. Selection process starts by calculating a tuple hash $\mathcal{H}(t)$using equation 11 such that $\mathcal{H}(t)$uniquely identifies each record in$\mathcal{A}$.

Figure 5. Pseudo-code for selecting the positions to store watermark bits.

$$\mathcal{H}(t) = Hash(\mathcal{K}_s||\mathcal{K}_d(t)||\mathcal{K}_s) \qquad (11)$$

Where, $\mathcal{K}_s$ is a secret key selected by the owner, || is the concatenation operator, hash(.) is a cryptographic hash function [22] and $\mathcal{K}_d(t)$) is the derived primary key of the tuple t [6].Equation12 calculates $\mathcal{K}_d$ for each tuple.

$$\mathcal{K}_d(t) = MSB(t.\mathrm{A}_1)||MSB(t.\mathrm{A}_2) \qquad (12)$$

Where,$MSB(t.A_1)$ and $MSB(t.A_2)$ are the most significant bits (MSB) of the attributes in $\mathcal{A}$.The attributes that are included in reducts contribute in creation of derived primary key. These attributes contain crucial information that cannot be deleted from a database. In case of any perturbations in these attributes, only Least Significant Bits (LSBs) will change, abstaining MSBs. MSB space is assumed to be a domain where minor changes on the collection items have a minimal impact on the MSB labels. Any further attempt to change MSBs, will affect the usefulness of the data. Hence, derived primary key of all tuples will remain same even after slight modifications.

If the modulus of $\mathcal{H}(t)[1:20]$ with$1/\gamma$yields zero, then the tuple is selected for inserting watermark bit and the tuple's $W\_Status(t)$ is set to 1 (lines 5, 6). This process serves two purposes. One, it scatters the watermark throughout the entire $\mathcal{A}$ thus making it difficult for an attacker to locate the watermark bits. Secondly, the use of a secure hash function, secret parameters $\mathcal{K}_s$andγenhances security by concealing the identity of the watermarked tuples

from an intruder. γ is selected such that $\gamma * N_t$ is almost equivalent to N_w so that the watermark is scattered throughout the database.Within a selected tuple, we choose target attributes with index $AIndex(t)$and$(AIndex(t) + 1)$tocarry the watermark bits (line 7). Our scheme selects two attributes per selected tuple for embedding watermark bits. This process adds another level of obfuscation by ensuring that the same attribute is not always chosen across different selected tuples for the purpose of embedding.

B. Embed Watermark: The LSBs of selected attributes serve as the potential locations for carrying different bits of the watermark. A single bit is embedded in each selected location and the process stops when all watermark bits $\mathbb{W}_1 .. \mathbb{W}_{N_w}$are embedded. There are very rare chances of change in reducts and/or rules due to embedding of watermark bit into LSB of selected non-reduct attributes. After embedding watermark bits, the reducts and rules are verified. We ignore those positions which reflect change in rules and/or reducts after embedding a watermark bit.

Once the DS, $\mathcal{A}$ is armed with its signature watermark $\mathbb{W}_s$ (now referred as $\mathcal{A}_w$), 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 suspected database as $\mathcal{A}^{'}$.

4.3.Watermark Extraction

The watermark extraction process *Extract_Watermark(.)*is outlined in figure 6. This process is responsible for extracting the embedded watermark from a suspected watermarked database$\mathcal{A}^{'}$. The proposed watermark extraction technique is blind as it does not require the original database for extracting the watermark from the suspected database.

The subroutine *Extract_Watermark(.)* starts by calling *Select_Positions*(.) to re-calculate the original embed locations (line 1). It then extracts the watermark bit from the LSB of each earmarked attribute of selected tuple (line 4, 5).

4.4.Decision Generation

This step generates the decision that whether the database is tampered. Tamper detection takes place at the receiver side. Process *Extract_Watermark*(.)is invoked to extract the watermark $\mathbb{W}_x$. Next, process *Create_Watermark*(.) is invoked to re-create the watermark $\mathbb{W}_g$from the suspected database $\mathcal{A}'$.

Figure 6. Pseudo-code for extracting the watermark from $\mathcal{A}'$

Let us consider the following cases that may arise:

1. There were no integrity attacks. If neither the data nor the watermark were changed, then $\mathfrak{W}_g == \mathfrak{W}_s$ and $\mathfrak{W}_x == \mathfrak{W}_s$.Thus, the two watermarks will match$(\mathfrak{W}_g == \mathfrak{W}_x)$.

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.* $\mathfrak{W}_g \neq \mathfrak{W}_s$. Thus, $\mathfrak{W}_x \neq \mathfrak{W}_g$ and the tampering event will surely be detected.

3. The content of the DS was changed, but that did not result in corresponding change in the decision rules as well as change in the embedded watermark. The change in DS does not alter values in the embed watermark positions. Hence, $\mathfrak{W}_g = \mathfrak{W}_s = \mathfrak{W}_x$. Since the desired information encapsulated within the DS was not affected despite tampering, this result is acceptable.

4. The raw data in the DS was not changed but the positions where watermark is embedded was tampered. In this case, the reducts and rules will remain the same. Hence $\mathfrak{W}_g = \mathfrak{W}_s$but due to tampering, $\mathfrak{W}_x$changes. Hence,$\mathfrak{W}_x \neq \mathfrak{W}_g$, and the tampering is detected.

5. Both reducts/rules as well as the embedded watermark was 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 inserted watermark respectively, turns out to be exactly the same. Hence $\mathfrak{W}_x \neq \mathfrak{W}_g$and 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.

5. SECURITY ANALYSIS

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 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 and/or the rules and/or their support values, then it will result in a new regenerated watermark $\mathbb{W}_g$. The attacker will now try to make changes to the embedded watermark so that when it is extracted, $\mathbb{W}_x$ becomes equal to$\mathbb{W}_g$. 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 modulus and a cryptographic hash function, the output values are completely randomized with uniform distribution. The watermark of length N_w is embedded into $\gamma * N_t$ selected tuples. Thus, the probability of finding all the $\gamma * N_t$watermarked tuples is given as:

$$P_t = \frac{\gamma * N_t}{N_t} * \frac{(\gamma * N_t) - 1}{(N_t - 1)} * \frac{(\gamma * N_t) - 2}{(N_t - 2)} * \dots\dots * \frac{1}{(N_t - \gamma * N_t + 1)}$$

$$= \left(\frac{1}{{}^{N_t}C_{\gamma * N_t}}\right) \qquad (13)$$

Without knowing set of candidate attributes A_β, Mallory will select A_F as a set of all attributesthat can be in candidate attributes set; such that $A_\beta \subseteq A_F$. For each tuple, two different attributes are selected out of β attributes.An earmarked attribute can be chosen in $1/\beta_F$ways.Thus, the probability of correctly identifying watermarked locations of all the $\gamma * N_t$tuples is given as:

$$P_l = \left\{\left(\frac{\gamma * N_t}{N_t}\right) * \left(\frac{2}{(\beta_F)(\beta_F - 1)}\right)\right\} * \left\{\frac{(\gamma * N_t) - 1}{(N_t - 1)} * \left(\frac{2}{(\beta_F)(\beta_F - 1)}\right)\right\} * \dots$$
$$* \left\{\left(\frac{1}{(N_t - \gamma * N_t + 1)}\right) * \left(\frac{2}{(\beta_F)(\beta_F - 1)}\right)\right\}$$

$$P_l = \frac{1}{\left({}^{N_t}C_{\gamma N_t}\right)} * \left(\frac{2}{\beta_F(\beta_F - 1)}\right)^{\gamma N_t} \qquad (14)$$

Substituting $\gamma = 1/3, \beta_F = 10, N_t = 6000$ in equation 14, we get $P_l = 0.26\, e^{-5054}$which is negligible. Hence, it is impossible for the Mallory to correctly identify

the watermarked positions. We studied the effect of $\beta_F$on $P_l$and recorded the values obtained in Table 3.The values reveal that as a number of candidate attributes increases, probability decreases further. Similarly, we observed the change in behaviour with respect to N_t and recorded in Table 4. The probability is inversely proportional to γ and N_t, as it decreases with the increase in γ and N_t.

Table 3. Effect of β_F on P_l $(* e^{-2000})$ at different values of γ

Table 4. Effect of β_F on $P_l(* e^{-2000})$ at different values of N_t

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 is ignored.

Scenario 2: The attacker has tampered the embedded watermark and will now try to work backwards to alter the DS so that the regenerated watermark $\mathbb{W}_g$ becomes the same as the watermark extracted $\mathbb{W}_x$ after tampering it.The only option left is a brute force attack. An attacker will try out all possible values of reducts in-order to change the rules and thus, the watermark. For larger databases with considerable number of reducts; this is close to impossible to get desired watermark.

***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 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 a new watermark by sheer chance is the same as probability that $N_w$bits of both watermark matches. Each bit can be equal to 0 or 1 and is independent of other bits. Thus, probability that all N_w bits matches is $\left(\frac{1}{2}\right)^{N_w}$. For the watermark with $N_w = 256$, we get$P_e = 2.9387e^{-39}$, which is negligible.

6. EXPERIMENTAL RESULTS AND ANALYSIS

Rough Set Theory (RST) is a powerful tool for conducting an approximate analysis of

DS and extracting classification rules that guide in discerning objects. These rules can be applied to label new objects. We used Rosetta tool version 1.4.40 to implement RST. This software helps us analyze rough sets and their behavior. We have performed our experiments on the datasets enlisted in table 5 available as sample databases in Rosetta software. These databases contain varying number of records to aid in studying the effect on number of records.

Table 5 Datasets used in experiments

6.1. *Extraction of reducts and rules*

In our experiments, we applied Johnson algorithm for reduction of the dataset and full indiscernibility for the generation of rules [20-21].Table 6 shows the results of applying Johnson's and genetic algorithms for obtaining reducts and rules on different decision systems. Since our objective is to detect the changes made in the pivotal informative attribute values, we prefer Johnson's algorithm with object type discernability to get rules based on each object/record. This aids us in localising the tampered object. Results show that number of rules generated in case of Johnson's algorithm is least. For the proposed work, we aim at minimizing the watermark bits. Thus, we follow Johnson's algorithm to extract rules from the decision system.

Table 6. Comparison of Johnson's and Genetic algorithms to obtain reducts and rules.

6.2. Generation of Watermark

We prepare the watermark from the extracted reducts and rules and embed them into the decision system. The length of the watermark is to be minimal; to embed watermark robustly. We tried GZIP[24], BZIP2[25],LZMA2[26], and PPMd[27-28] compression algorithms to minimize the watermark and recorded the result in Table 7. Records clearly show that PPMd has higher compression ratio and it increases with increase in input size. For large size databases, PPMd is preferred [29].We have used PPMd in our proposed scheme to achieve high compression ratio.

Table 7. Comparison of various compression techniques on different datasets.

We next analyzed the integrity of the watermarked DS against various attacks.

6.3. Integrity analysis

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

A. Subset Addition Attack: In this attack, by adding some spurious tuples in DS, Mallory hopes that the watermark embedded in DS remains untouched. However, the added tuple contains the attribute that are included in reducts and therefore, may contribute in formation of rules. Thus, these modifications to DS is bound to alter its signature. The graph in figure 7 shows that on adding only 10% of total records, there is as much as 7.8% change in the recreated rules N_{runew} from perturbed database. As the percentage of added records increases, change in rules N_{runew} and reducts N_{rnew} increases. We have shown the results till 50% alteration in database, as beyond this the database become useless.

i. ***Characterize and Localization:*** We deliberately added spurious records in DS and record the results in table 8, after regenerating the rules from modified DS. Table 8 shows that there is an increase in the number of regenerated rules as the perturbation in number of records increases. Thus, we can characterize this attack by analyzing the number of rules regenerated from the perturbed DS. Additionally, comparing the values of the attributes included in the reducts with those of newly added rules, we can localize the record that is added.

Figure 7. Simulation results of subset addition attack showing change in percentage of regenerated reducts and rules with the increase in the addition of tuples in DS.

Table 8. Number of rules regenerated Nru with perturbations in the number of records.

B. Subset Alteration Attack: Mallory tries to modify the DS by randomly altering some data bits such that the data is not completely rendered useless. To simulate this kind of attack,

we alter the attributes that participated in reducts of DS within usability constraints and recreate the watermark using C*reate_Watermark(.)* procedure. The graph in figure 8shows significant changes in the recreated rules when compared with original rules. There is 49.02% change in recreated rules on altering 50% of tuples. Even though, the reducts are left unperturbed the technique detects change in recreated watermark. Thus, our technique is fragile to this attack.

Figure 8. Simulation results of subset alteration attack showing change in percentage of regenerated rules and reducts.

Table 9. Percent change in number of rules regenerated Nru with perturbations in the reducts and non-reducts attributes of various records.

The same experiment is repeated by altering the attributes that are not included in reducts. This does not affect the recreated rules and reducts. However, this may alter the extracted watermark. Table 9 shows the change in number of rules regenerated when the reduct and non-reduct attributes are altered. The graph in figure 9 shows the changes in extracted watermark by altering attributes that are not in reducts of random tuples. Only selected tuples participate in concealing watermark, thus the resulting change in extracted watermark is not uniform.

Figure 9. Simulation results showing change in percentage of extracted watermark with altering attributes other than in reducts.

i. ***Characterize and Localization:*** We deliberately modified certain records in DS and record the results in table 8, after regenerating the rules from modified DS. Table 8 shows that there is no change in the number of regenerated rules with the increase in perturbation in number of records. However, the watermark extracted and watermark regenerated will not match as either the content of regenerated rules will differ or the watermarked positions will differ. Thus, we can characterize this attack. In case the attributes participating in reducts are altered; we may localize that attribute by analysing the values of the reducts attributes in regenerated rules with those in extracted ones.

C. Subset Deletion Attack: Mallory tries to delete records from the DS with an intention of altering the DS.This may also affect the embedded watermark. Thus, both recreated $\mathfrak{W}_g$ and

extracted watermark $\mathbb{W}_x$may change. The graphical result shown in figure10 reveals that even if 10% records are deleted, the percentage change in rules alters significantly upto 11.76%. Thus, one can easily claim that the DS was tampered with.

Figure 10. Simulation results in case of Subset Deletion attack

i. ***Localization:*** By comparing the rules extracted from the embedded watermark with the regenerated rules, we can identify the type of attack. Table 8 shows that there is decrease in the number of regenerated rules as more number of records is deleted. Thus, we can characterize this attack. Further, by comparing the rules obtained from extracted watermark along with their support with regenerated rules, we can easily identify the missing values of the reducts.

7. CONCLUSIONS

In this paper, we proposed a novel fragile watermarking technique which detects integrity losses in the knowledge regarding the ability to discern between objects in DS. This is achieved by preparing a signature of this core information securely, by creating reducts and rules derived from the DS using RST. The created 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. The proposed technique employs three security levels to complete the entire watermarking process. Firstly, watermark is prepared securely using a secret key making difficult for an attacker to crack the watermark. Secondly, tuples where a watermark is embedded is chosen using a secret parameter. Lastly, the selection of attributes within earmarked tuples is done securely using other secret parameters. Further, the fragility of the proposed scheme is experimentally proven against various attacks. The proposed technique detects, characterizes and localizes the modifications made to the DS.

We have proposed a scheme on tamper detection of the latent information in DS databases. This can be extended to provide the protection of entire DS. Further the technique can be enhanced to provide recovery of lost/perturbed values.

FIGURES

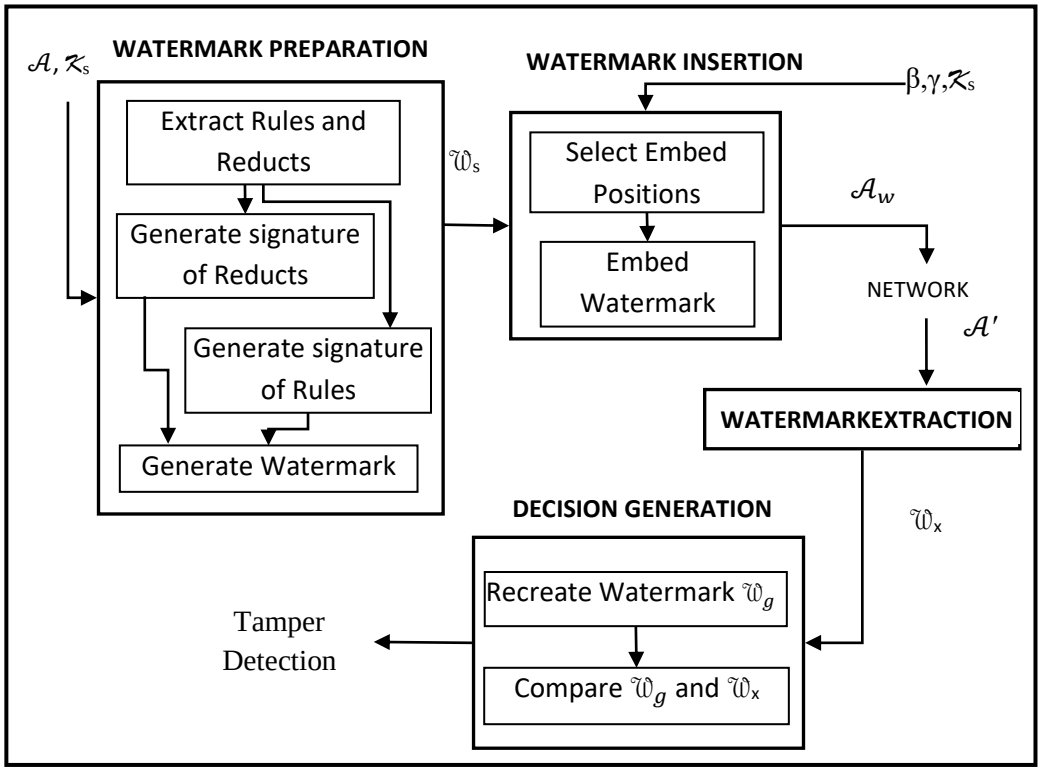

Figure 1.Block diagram of a proposed fragile watermarking model for Decision System.

Create_Watermark(.)

*Input:*Dataset $\mathcal{A}$, Secret key $\mathcal{K}_s$

Output: Watermark$\mathbb{W}_s$

1. Extract reducts and rules of Dataset$\mathcal{A}$.
2. Create reducts signature Sig_A by using template given in figure 4(a).
3. Create signature of decision rules Sig_B by using the template given in figure 4(b).
4. Calculate Watermark$\mathbb{W}_s = Compress(Sig_A \parallel Sig_B \parallel \mathcal{K}_s)$

Figure 2. Pseudo-code for creating a watermark $\mathbb{W}_s$ from dataset $\mathcal{A}$

Signature of ReductsSig_A	Signature of RulesSig_B	Secret Key$\mathcal{K}_s$

Figure 3. Template of watermark $\mathbb{W}_s$

***Sig_A*: Signature for 1..N Reducts**													
R_1				…	R_i				…	R_N			
Length	**Index of Attributes**			…	**Length**	**Index of Attributes**			**...**	**Length**	**Index of Attributes**		
L_1	$I(A_{1,1})$	...	$I(A_{1,L_1})$	...	L_i	$I(A_{i,1})$	...	$I(A_{i,L_i})$	...	L_N	$I(A_{N,1})$	...	$I(A_{N,L_N})$

Fig. 4(a): Template for preparing the signature of Reducts

***Sig_B:* Rules for equivalence class E_{i1} of Reduct R_i**												..	E_{i,N_q}
Antecedent						**All Consequents**							
						Decision-class, Support							
N_{Ru}	$V_x(A_{i,1})$	...	$V_y(A_{i,j})$	...	$V_z(A_{i,L_i})$	d_1	Sup_1	...	d_n	Sup_n	...	...	

Fig. 4(b): Template for generating the signature of Rules

Figure 4. Template for generating *Sig_A* and *Sig_B*

***Select_Positions*(.)**

*Input:*Dataset $\mathcal{A}$,$\mathcal{K}_s, \gamma, \beta$.
Output:$W\_Status(t), AIndex(t)$

1. **For** each tuple t in $\mathcal{A}$
2. Initialize $W\_Status(t)$=0
3. Calculate $\mathcal{H}(t)$using equation 11
4. Assign$\mathcal{H}1(t) = \mathcal{H}(t)[1:20]$ and $\mathcal{H}2(t) = \mathcal{H}(t)[1:30]$
5. **If** $(\mathcal{H}1(t) mod\ 1/\gamma == 0)$
6. Set $W\_Status(t)$ =1
7. $AIndex(t) = \mathcal{H}2(t) mod\ \beta$
8. **End if** (line 5)
9. **End for**(line 1)

Figure 5.Pseudo-code for selecting the positions to store watermark bits.

Extract_Watermark(.)

Input: Suspected watermarked database $\mathcal{A}'$, W_Status(t), $AIndex(t)$

Output: Extracted watermark $\mathfrak{W}_x$

1. Call *Select_Positions(.)* inscribed in figure 5.
2. Initialize *count* =1.
3. **For** each tuple $t\varepsilon\,\mathcal{A}'$
4. **If** $(W\_Status(t) == 1)$
5. Extract the watermark bit $\mathfrak{W}_x[count]$ from theLSB of $A_{AIndex(t)}(t)$
6. Increment *count* by 1.
7. Extract the watermark bit $\mathfrak{W}_x[count]$ from theLSB of $A_{(AIndex(t)+1)}(t)$
8. Increment *count* by 1.
9. **If** $count > lengthofwatermarkN_w$
10. **Exit for**(line 3)
11. **End if** (line 9, line 4)
12. **End for** (line 3)

Figure 6. Pseudo-code for extracting the watermark from $\mathcal{A}'$

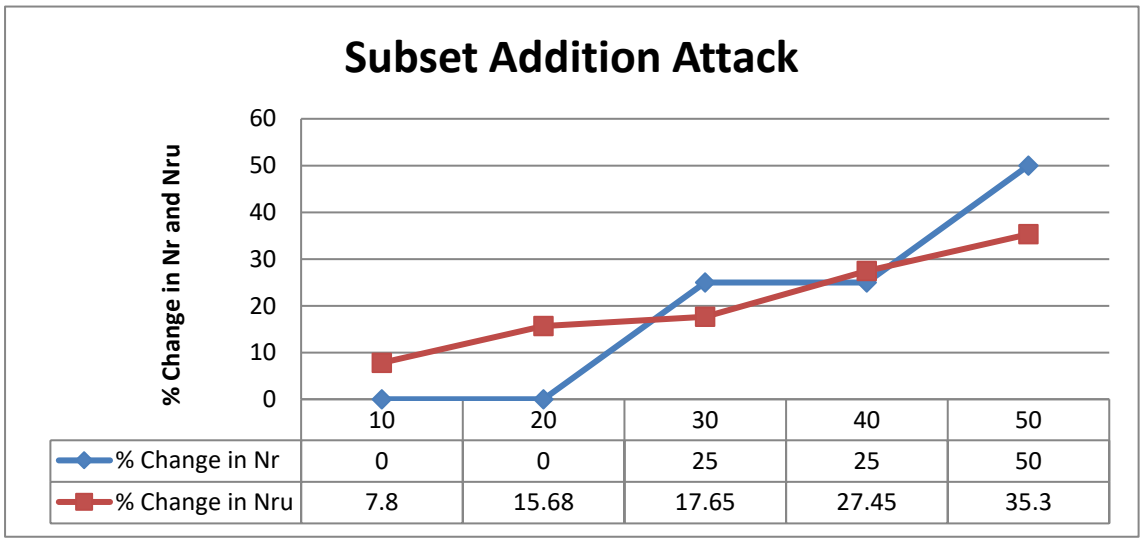

Figure 7. Simulation results of subset addition attack showing change in percentage of regenerated reducts and rules with the increase in the addition of tuples in DS.

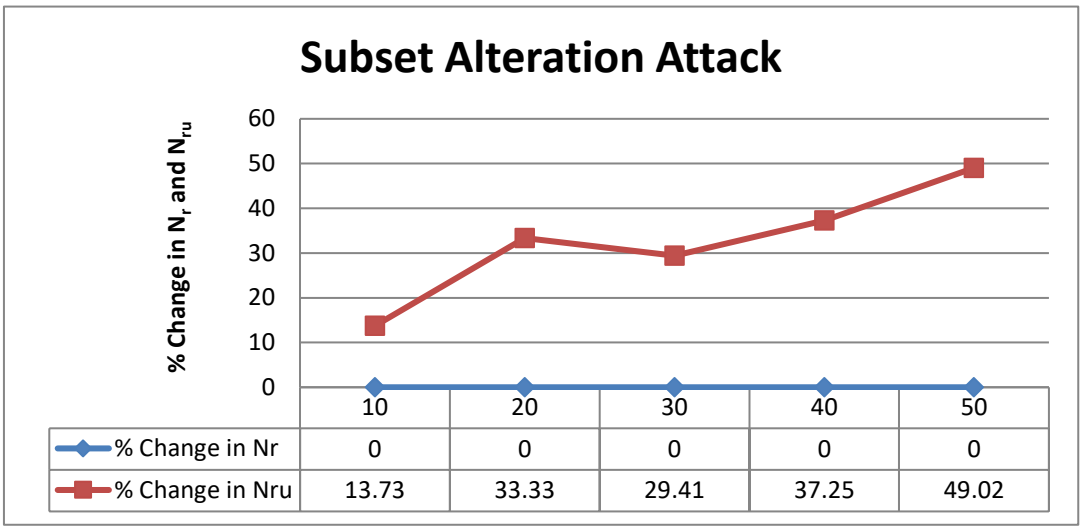

Figure 8. Simulation results of subset alteration attack showing change in percentage of regenerated rules and reducts.

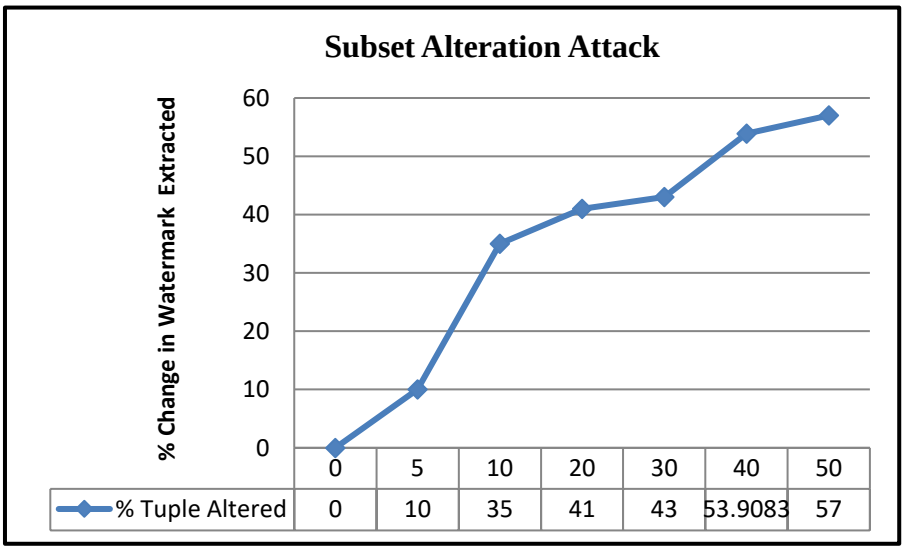

Figure 9. Simulation results showing change in percentage of extracted watermark with altering attributes other than in reducts.

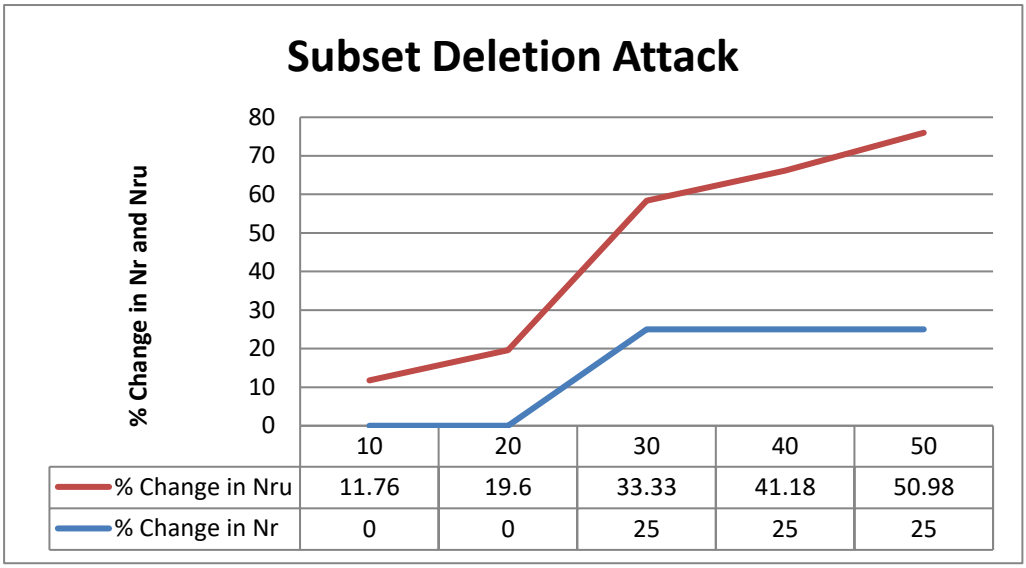

Figure 10. Simulation results in case of Subset Deletion attack

TABLES

Table.1. Table of Symbols

Symbol	Description
$\mathcal{A}$	Decision support based dataset
$\mathcal{A}_w$	Watermarked Data set
N_t	Total number of tuples in $\mathcal{A}$
N_a	Total number of attributes in $\mathcal{A}$
K_s	Secret key decided by owner of the dataset
γ	Fraction of tuples chosen for embedding watermark
$\mathbb{W}_s$	Signature watermark prepared
β	Fraction of attributes selected for embedding watermark
$H(t)$	Hash of t^{th} tuple
L_w	Length of the watermark $\mathbb{W}_s$

Table 2: Obtaining reducts and rules from DS.

S.No	Sample Dataset	Number of records N_t	Number of attributes N_a	Number of Reducts N_r	No. of distinct attributes in Reducts	Number of Rules N_{ru}
1	Australian	690	15	18	08	549
2	Cleveland dataset	303	15	12	07	235
3	Flag	24	11	01	06	12
4	HSV	61	12	04	04	51

Table 3. Effect of β_F *on* P_l $(* e^{-2000})$ at different values of γ

	$\gamma = 1/2$	$\gamma = 1/3$	$\gamma = 1/5$
β_F **=5**	$0.64e^{-2804}$	$0.23e^{-1656}$	$0.9e^{-502}$
β_F **=10**	$0.77e^{-4901}$	$0.26e^{-3054}$	$0.15e^{-1340}$
β_F **=15**	$0.34e^{-5941}$	$0.7e^{-3748}$	$0.1e^{-1756}$
β_F **=20**	$0.52e^{-6707}$	$0.2e^{-4258}$	$0.52e^{-2063}$

Table 4. Effect of β_F *on* $P_l(* e^{-2000})$ at different values of N_t

	N_t = 3000	N_t =6000	N_t =9000
β_F = 05	$0.3e^{173}$	$0.23e^{-1656}$	$0.14e^{-3485}$
β_F = 10	$0.3e^{-252}$	$0.26e^{-3054}$	$0.16e^{-5582}$
β_F = 15	$0.5e^{-873}$	$0.7e^{-3748}$	$0.74e^{-6623}$
β_F= 20	$0.28e^{-1128}$	$0.2e^{-4258}$	$0.11e^{-7388}$

Table 5 Datasets used in experiments

S.No	Dataset Name	No of tuples in training data set	No of attributes excluding decision attribute	No of decision attribute	No of decision set
1.	Australian	690	14	1	2
2.	Cleveland	303	14	1	2
3.	Flags	24	10	1	2
4.	HSV	61	11	1	4
5.	Iris	150	4	1	3

Table 6. Comparison of Johnson's and Genetic algorithms to obtain reducts and rules.

S.No	Sample Dataset	Algorithm Applied	Number of Reducts Generated N_r	Number of Rules generated N_{ru}
1	Australian	Johnsons's	18	549
		Genetic	608	10384
2	Cleveland dataset	Johnsons's	12	235
		Genetic	472	5572
3	Flag	Johnsons's	01	12
		Genetic	56	130
4	HSV	Johnsons's	04	51
		Genetic	56	567
5.	IRIS	Johnsons's	7	77
		Genetic	11	246

Table 7. Comparison of various compression techniques on different datasets.

S.No	Sample Dataset	Compression Algorithm	Number of Rules generated	Length of Binary Stream before compression	Length of Binary Stream after compression	Compression Ratio
1	Australian	GZIP	549	12538	1824	7:1
		BZIP2			1555	8:1
		LZMA2			1916	6:1
		PPMd			1575	8:1
2	Cleveland dataset	GZIP	235	7544	929	8.12
		BZIP2			823	9.16
		LZMA2			1008	7.48
		PPMd			893	8.45
3	Flag	GZIP	24	660	242	2.72
		BZIP2			255	2.58
		LZMA2			342	1.93
		PPMd			298	2.21
4	HSV	GZIP	51	1235	253	4.88
		BZIP2			237	5.21
		LZMA2			342	3.61
		PPMd			298	4.14
5	IRIS	GZIP	77	3669	392	9.36
		BZIP2			348	10.54
		LZMA2			446	8.2
		PPMd			459	7.99

Table 8.Number of rules regenerated Nru with perturbations in the number of records.

% Records perturbed	0	10	20	30	40	50
Subset Addition (N_{ru}')	51	55	59	60	65	69
Subset Deletion (N_{ru}')	51	45	41	34	30	25
Subset Alteration (N_{ru}')	51	51	51	51	51	51

Table 9.Percent change in number of rules regenerated Nru with perturbations in the reducts and non-reducts attributes of various records.

% Change in Records	**10**	**20**	**30**	**40**	**50**
% Change in rules with alteration in Non-Reducts Attributes	0%	0%	0%	0%	0%
% Change in rules with Alteration in Reducts Attributes	13.73	33.33	29.41	37.25	49.02

REFERENCES

[1] Bertino.E, Sandhu. R: Database Security—Concepts, Approaches, and Challenges, IEEE Trans. on Dependable and Secure Computing, 2(1), 2-19 (2005).

[2] Brown I.E: The evolution of anti-circumvention law, International Review of Law, Computers and Technology, 20(3), 239-260(2006)

[3] Agrawal R., Haas.P.J, Kiernan.J: Watermarking relational data: framework, algorithms and analysis, Very Large Databases Journal, 12(2), 157-169(2003).

[4] Sion. R.,,Atallah.M. , Prabhakar.S: Rights protection for relational data, IEEE Trans. on Knowledge and Data Engineering , 16(12), 1509-1525(2004).

[5] Ifthikar S., Kamran M. and Anwar Z.: RRW-A robust and reversible watermarking technique for relational Data, IEEE Trans. on Knowledge and Data Engineering, 27(4), 1131-1145(2015).

[6] Khanduja.V.,Verma.O.P., Chakraverty.S: Watermarking Relational databases using Bacterial Foraging Algorithm, Multimedia tools and Applications, Springer, 74(3), 813-839 (2015), DOI: 10.1007/s11042-013-1700-9.

[7] Sion. R., Atallah. M., Prabhakar.S.: Rights protection for categorical data, IEEE Trans. on TKDE, 17(7), 912-926 (2005).

[8] Khanduja.V., Chakraverty.S., Verma.O.P.: A Scheme for Robust Biometric Watermarking in Web Databases for Ownership Proof with Identification, International conference on Active Media Technology, Poland, LNCS 8610, 212-215 (2014).

[9] Li.Y., Guo.H., Jajodia.S.: Tamper Detection and Localization for Categorical Data Using Fragile Watermarks, Proceedings of the 4th ACM Workshop on Digital Rights Management, Washington DC, USA, 73-82(2004).

[10] Guo.H., Li.Y., Liu.A., Jajodia.S.: A fragile watermarking scheme for detecting malicious modifications of database relations, Information Sciences, Elsevier, vol. 176, 1350–1378 (2006).

[11] Khanduja.V., Chakraverty.S., Verma.O.P., Rakshita, Goel.S.: A Robust Multiple Watermarking Technique for Information Recovery, Proceedings of the IEEE Int. Conf. on Advanced Computing, India, 250-255(2014).
[12] Khanduja.V., Chakraverty.S., Verma.O.P.: Enabling Information Recovery with Ownership using Robust Multiple Watermarks, Journal of Information Security and Applications, Elsevier, 80-92(2016).
[13] Schäler.M., Schulze.S., Merkel.R., Saake. G., Dittmann.J.: Reliable Provenance Information for Multimedia Data Using Invertible Fragile Watermarks, BNCOD, LNCS 7051, 3–17(2011).
[14] Khanduja.V., Chakraverty.S., Verma.O.P.: Robust Watermarking For Categorical Data, Proceedings of the IEEE Int. conf. on Control, Computing, communication and materials, Allahabad, 174-176(2013).
[15] Bhattacharya.S., Cortesi.A.: Database authentication by distortion free watermarking, Proceedings of the Int. conf. on Software and Data Technology, SciTePress, 219-226(2010).
[16] Pawlak. Z.:Rough Sets, Int. Journal of Computer and Information Sciences, vol. 11, 341-356(1982).
[17] Pawlak.Z.: Rough set theory and its applications, Journal of Telecommunication and Information Theory, vol.3, 7-10(2002).
[18] Komorowski, Polkowski.L., Skowron.A.: Rough Sets: A Tutorial, Rough Fuzzy Hybridization- A new Trend in Decision Making, Springer, 3-98(1998).
[19] Pawlak Z.: Rough Sets: Theoretical Aspects of Reasoning about Data,,Kluwer Academic Publishers, Dordrecht(1991).
[20] Hoa, Sinh.N., Nguyen Hung Son.: Some efficient algorithms for rough set methods, Proceedings IPMU. Vol. 96(1996).
[21] Torgeir R. Hvidsten .: A tutorial-based guide to the ROSETTA system: A Rough Set Toolkit for Analysis of Data, 2nd Ed(2010).
[22] Khanduja, V., Chakraverty, S., Verma, O.P.: Ownership and Tamper Detection of Relational Data: Framework, Techniques and Security Analysis, Editors: C.W. Ahn, M. Ali, M. Pant, Embodying Intelligence in Multimedia Data Hiding, Vol. 5, Ch2, GCSR, 21-36 (2016),DOI: 10.15579/gcsr..
[23] Schneier B., Applied Cryptography, protocols, algorithms and source code in C, 2nd edn. Wiley-India, 2008.
[24] Deutsch, L. Peter:GZIP file format specification version 4.3(1996).
[25] BZIP@.... J. Seward, The bzip2 program (1996), http://www.bzip.org/. Accessed 10 September 2016
[26] Igor Pavlov.: 7z format, http://www.7zip.org/7z.html. Accessed 10 Spetember 2016
[27] Shkarin, Dmitry: PPM: One step to practicality, Proc. 2002 Data Compression Conference. 202-211. doi: 10.1109/DCC.2002.999958
[28] W. J. Teahan: The ppmd+ program, (1997) ftp://ftp.cs.waikato.ac.nz/pub/compression/ppm/ ppm.tar. gz . Accessed 21 April 2016.
[29] Gupta.A, Bansal.A, Khanduja.V.: Modern lossless compression techniques: Review, Comparison and Analysis, IEEE int. conf. on Electrical, Computer and Communication Technologies, 1083-1090(2017) doi: 10.1109/ICECCT.2017.8117850