Inf Syst Front (2013) 15:35–53
DOI 10.1007/s10796-010-9252-2

Two-stage database intrusion detection by combining multiple evidence and belief update

Suvasini Panigrahi · Shamik Sural · Arun K. Majumdar

Published online: 11 August 2010

Abstract Insider threats have gained prominence and pose the most challenging threats to a database system. In this paper, we have proposed a new approach for detecting intrusive attacks in databases by fusion of information sources and use of belief update. In database intrusion detection, only intra-transactional features are not sufficient for detecting attackers within the organization as they are potentially familiar with the day-to-day work. Thus, the proposed system uses inter-transactional as well as intra-transactional features for intrusion detection. Moreover, we have also considered three different sensitivity levels of table attributes for keeping track of the malicious modification of the highly sensitive attributes more carefully. We have analyzed the performance of the proposed database intrusion detection system using stochastic models. Our system performs significantly better compared to two intrusion detection systems recently proposed in the literature.

S. Panigrahi (✉)
School of Computer Engineering, KIIT University, Bhubaneswar, India
e-mail: spanigrahifcs@kiit.ac.in, suvasini26@gmail.com

S. Sural
School of Information Technology, Indian Institute of Technology Kharagpur, Kharagpur, India
e-mail: shamik@sit.iitkgp.ernet.in

A. K. Majumdar
Department of Computer Science & Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India
e-mail: akmj@cse.iitkgp.ernet.in

Keywords Database security · Dempster–Shafer theory · Bayesian learning · Intrusion detection · Insider attack · Suspicion score

1 Introduction

Concern regarding the security of databases has become more crucial than ever before in all information infrastructures. According to a computer crime and security survey conducted by the Computer Security Institute (CSI) (Gordon et al. 2009) in 2005, approximately 45% of the inquired entities who responded have reported increased unauthorized access to information. The 2007 CSI computer crime and security survey (Richardson 2009) proclaimed financial application fraud as the leading cause of financial loss and found it had more than doubled as compared to the loss estimated in the previous year. These figures show the growing sophistication and stealth of information attacks in databases. In addition to the substantial financial losses, these attacks can also tarnish the reputation of organizations, cause loss of customer confidence, and even lead to litigations.

Traditional database security mechanisms provide security features such as authentication, authorization, access control, data encryption and auditing. However, they are often found to be inadequate in satisfying the security needs of modern information systems. Despite the use of these preventive measures, data contained in a database can be corrupted by authorized insiders with malafide intent, or outside attackers who have assumed an insider's identity. Moreover, in databases, some of the attributes are more sensitive to malicious modifications as compared to others. Since all attacks cannot be prevented, the development of effective database intrusion detection systems (DIDSs) is essential for protecting sensitive and proprietary information in databases, and yet it remains an elusive goal and a challenging problem.

Anyone within or outside an organization could be an intruder. Intrusion can be classified as outside or inside based on the source from which it occurs. In case of outside intrusion, malicious transactions are executed by unauthorized users from outside the organization, who may gain access to the database by exploiting system vulnerabilities. The person who intrudes the system in such a manner is called an outsider. In this type of intrusion, the intruder may not be aware of the security layout of the organization and the database schema. An inside intrusion (Furnell 2004) is one in which unauthorized database transactions are carried out by authorized users, within the organization. A person who intrudes from within an organization is called an insider. These attacks are particularly difficult to defend against as the intruders are authorized users of the system and may have certain access rights to data and resources (Furnell 2004). Besides, insiders are potentially familiar with a part of the database schema along with the security setup the organization. Once such intruders manage to get the authentication information of a normal user, they can submit transactions similar to the genuine ones. Inside intrusions can remain undetected for a long time and thereby cause serious damage to database systems. Murray (2005) has found that the primary security threats come from internal misuse rather than from external attacks. Thus, insider attacks bring the most challenging threats to a database system and for this reason we focus on identifying this type of intrusion.

The attributes corresponding to a single transaction are known as intra-transactional and attributes related to multiple transactions are called inter-transactional. An intrusion detection system (IDS) which detects intrusion only based on intra-transactional features like query type, accessed table name, accessed attribute name, transaction location and transaction time, cannot identify the insider attacks as malicious. Therefore, only intra-transactional features are not sufficient in database intrusion detection. When an attacker requests multiple transactions, it is possible to identify inter-transactional deviation (which attributes are accessed after which attributes, which types of queries are invoked after which types of queries, time gap between transactions by the same user, etc.) even though the individual transactions are quite similar to the normal transactions. Thus, inter-transactional features should be considered as a significant part of intrusion detection.

It is well known that each user's normal profile/activity is unique and is represented using certain intra-transactional as well as inter-transactional features. The uniqueness of individual user's activity helps in identifying attempts by intruders masquerading as genuine users by capturing their deviation in current behavioral patterns from the normal profile. This is typically the case of inside intrusion. Thus, the basic idea of our approach is that as intruders are not totally familiar with the normal database access sequences, they usually show some intra-transactional as well as inter-transactional deviation in their database access. We use sequence alignment and spatio-temporal outlier detection and combine them using an extension of Dempster–Shafer theory to evaluate dissimilarity of any new database access sequence with respect to existing normal access sequences. A high score indicates potential abnormal activities in the system. It should be noted that inter-transactional features can be obtained only when multiple transactions are requested.

It has been found that the basic Dempster–Shafer theory does not quite well model evidences with a high degree of conflict. In this paper, we have, therefore, employed the Extended Dempster–Shafer theory (EDST) (Campos and Cavalcante 2003) for combining multiple evidences from the rules to compute an initial belief. Furthermore, the sensitivity levels of table attributes are also taken into consideration. This is done to keep track of the highly sensitive attributes more carefully, thus minimizing the overall loss suffered by the database owner due to intrusion.

The rest of the paper is organized as follows. Section 2 describes related work on database intrusion detection. We present the components of our proposed system named Two-Stage Database Intrusion Detection System (TSDIDS) in Section 3.1 followed by the intrusion detection steps in Section 3.2. In Section 4, we discuss the results obtained from various experiments. Finally, we conclude in Section 5 of the paper.

2 Related work

Research on intrusion detection has been conducted for nearly two decades, yet most of the existing intrusion detection systems are not capable of sufficiently identifying the presence of intrusions (Lunt 1996; Goan 1999). Existing work on intrusion detection has largely focused on network-based intrusion detection systems

(NIDSs) and host-based intrusion detection systems (HIDSs) (Hoglund et al. 2000; Giacinto et al. 2008; Hu et al. 2008; Triantafyllopoulos and Pikoulas 2002). In either case, the IDS looks for attack signatures, specific patterns that usually indicate malicious or suspicious intent. However, these IDSs do not work at the application level and hence are not capable of detecting intrusions in databases. An IDS working at the application level detects intrusions in the context of the application. It can use the semantics of the application to detect more subtle, stealth-like attacks such as those carried out by insiders. There are two main reasons for the requirement of database IDS (Kamra et al. 2007). Firstly, the actions considered as malicious in a database application are not necessarily malicious in network and operating system domain. Secondly, the IDS designed for networks and operating systems are not adequate to protect databases against insider attacks.

In spite of the significant role of databases in information systems, not enough attention has been paid to intrusion detection in database systems. A limited number of techniques have been proposed in the last few years for the detection of intrusion in databases. We briefly review some of them in this section.

Chung et al. (1999) present DEMIDS, a misuse detection system for relational database systems. It uses audit logs to derive user profiles which describe typical behavior of users and exploit them to detect misuse. The derived profiles are used to detect misuse behavior. This method assumes that the legitimate users show some level of consistency in using the database system. If this assumption does not hold, it results in a large number of false positives. Lee et al. (2002) designed a signature-based database IDS which works by matching new SQL statements against a known set of legitimate transaction fingerprints to detect database intrusions. However, generating the complete set of fingerprints for all database transactions and maintaining its consistency is a rigorous activity in case of large databases with enormous number of users. Moreover, if any of the legitimate transaction fingerprints are missing due to incomplete training data, it can cause many false alarms.

An interesting approach to mine user profile using query templates was suggested by Zhong and Qin (2004). They use constrained query template for improving the system effectiveness. Damiani et al. (2003) have suggested a robust single-server solution for remote querying of encrypted databases on untrusted servers by providing a hash-based method for database encryption.

Transaction level data dependency, a novel approach to represent genuine database access rules, was proposed by Hu and Panda (2005). In this approach, data dependency relationships among transactions are mined and this information is used to detect anomalies. Transactions not agreeable with any of the mined data dependency rules are identified as malicious. In every database, some of the attributes are considered more sensitive to malicious modification compared to others. Srivastava et al. (2006) have introduced the concept of attribute sensitivity in their work. They find weighted data dependency among data items and the transactions that do not follow these rules, are flagged as malicious.

Single-layered intrusion detection systems may raise a high number of false alarms. Wenhui and Tan (2001) proposed a two-layer mechanism to detect intrusions against web-based database services. They use the first layer to build historical profiles based on audit trails and other log data provided by the web server and the database server. A second layer is used to integrate the alarm context with the alarms generated from the first layer. Kamra et al. (2007) proposed a database IDS that has similarity with role-based access control (RBAC) model in profile granularity. Database log files are mined to generate user profiles that model normal user actions and is used to identify intrusions.

Till now, our discussion has been limited to user transactions. A new aspect of real-time database intrusion detection at the level of sensor transactions was proposed by Lee et al. (2000). They have employed the time semantics of temporal data objects to detect intrusions, which is unknown to the intruders. Barbara et al. (2002) suggested the use of Hidden Markov Model (HMM) for mining malicious data corruption.

From the above discussions, it may be noted that, Neural Network, Hidden Markov Model and Data Mining techniques are mostly used in the field of database intrusion detection. All these techniques aim to detect malicious transactions, specifically in databases, but an important problem in this field is to protect the database from well-formed but damaging transactions while limiting the generation of false alarms. Axelsson (2000) has pointed out that due to base-rate fallacy, the factor limiting the performance of an intrusion detection system is not the ability to identify intrusive behavior correctly but its ability to minimize false alarms. One of the motivations of our current research is to address this challenge.

It is well known that every user has a certain database access pattern, which are captured as rules by database intrusion detection systems. However, if these rules are static in nature, they become ineffective when a user develops new patterns of behavior over a period of time. Besides, new intrusion types, not known to the detection system, mostly go undetected. Thus, systems which do not combine multiple evidences or fail to learn the changing behavior of users, result in a large number of false alarms. Moreover, none of the existing systems consider intra-transactional features and inter-transactional features together for intrusion detection.

The normal database access profile of a user can be modeled at different granularity levels. Some possible transactional feature granularity levels include—query type, sensitive attributes, all attributes, operation on attributes and a combination of any of these. Nevertheless, the transactional feature used in each work as discussed above is static in nature. In reality, an IDS should be scalable enough to include all possible levels of transactional feature.

In this paper, we propose a database IDS which is designed with dynamic rules that support the selection of transactional feature granularity. Multiple evidences from these rules are combined using Extended Dempster–Shafer theory (EDST) (Campos and Cavalcante 2003). Once a transaction is found to be suspicious, belief update takes place based on its similarity with malicious or genuine transaction history using Bayesian learning. In addition, we consider both inter-transactional features (sensitive attribute access sequence as well as read/write operations on those attributes) and intra-transactional features (time gap between consecutive transactions by a user) together at a detailed level of granularity for effective intrusion detection. A preliminary version of this work has been reported in Panigrahi et al. (2009). To the best of our knowledge, this is the first ever attempt to develop a database intrusion detection system using information fusion and Bayesian learning.

3 Two-stage database intrusion detection system (TSDIDS)

The proposed database intrusion detection system combines evidences from current as well as past behavior of users. A number of dynamic rules are used to measure the deviation of each incoming transaction. An extension of Dempster–Shafer's theory (Campos and Cavalcante 2003) is applied for combining multiple such evidences and an initial belief is computed. First-stage decision making is done about each incoming transaction depending on its initial belief. If the initial belief is less than a certain lower threshold, the transaction is considered to be genuine. On the contrary, if the initial belief exceeds an upper threshold, then the transaction is declared as intrusive. In case the initial belief lies in between the two thresholds, the transaction is treated as suspicious and its suspicion score is further strengthened or weakened according to its similarity with malicious or genuine transaction history using Bayesian learning. The second stage decision making takes place in a similar manner based on the updated suspicion score of the transaction. TSDIDS may be abstractly represented as a 5-tuple $\langle U, P, \psi, \theta_{\mathrm{LT}}, \theta_{\mathrm{UT}} \rangle$ as shown in Fig. 1 where:

1. $U = \{U_1, U_2, ..., U_n\}$ is the set of users on which intrusion detection is performed
2. $P = \{P(U_1), P(U_2), ..., P(U_n)\}$ is the set of profiles, where each $P(U_k)$ corresponds to the profile of the user U_k. The profile representing user's normal behavior facilitates reliable intrusion detection by analyzing parameters of user's current behavior and comparing them with the user's profile. We have used multiple distinct intra-transactional as well as inter-transactional features collectively for profile representation since assimilation of multiple features enhances the performance of an IDS as compared to a single feature.

   We consider the following example of a transaction consisting of two queries *t1* then *t2* submitted to achieve a specific task.

   t1: *Select a, b from table T1 where c = 1*
   t2: *Update T4 set d = 10 where e = 1*

   where *a, b and c* are the attributes of table *T1* and *d and e* are the attributes of table *T4*. Suppose *a* and *d* are high sensitive attributes, *b* is medium sensitive and *c* and *e* are low sensitive attributes. The attribute sequence $< c, a, b >$ is considered to be an intra-transactional feature sequence as it is related to *t1* only. Similarly, we can define $< e, d >$ as an intra-transactional feature sequence using *t2*. The attribute sequence $< c, a, b, e, d >$ is considered to be an inter-transactional feature sequence where the first three attributes are taken from *t1* and the rest from *t2* sequentially.

   Each user profile $P(U_k)$ can be represented as a 7-tuple $\langle user\_ID, attrib\_ID\_seq, loc\_ID, time\_slot, table\_ID\_seq, attrib\_count, \rho \rangle$ where:

   – $user\_ID$: a number that identifies each user uniquely
   – $attrib\_ID\_seq$: attribute access sequence in a transaction. For this Eg. as discussed above, $attrib\_ID\_seq =< c, a, b, e, d >$

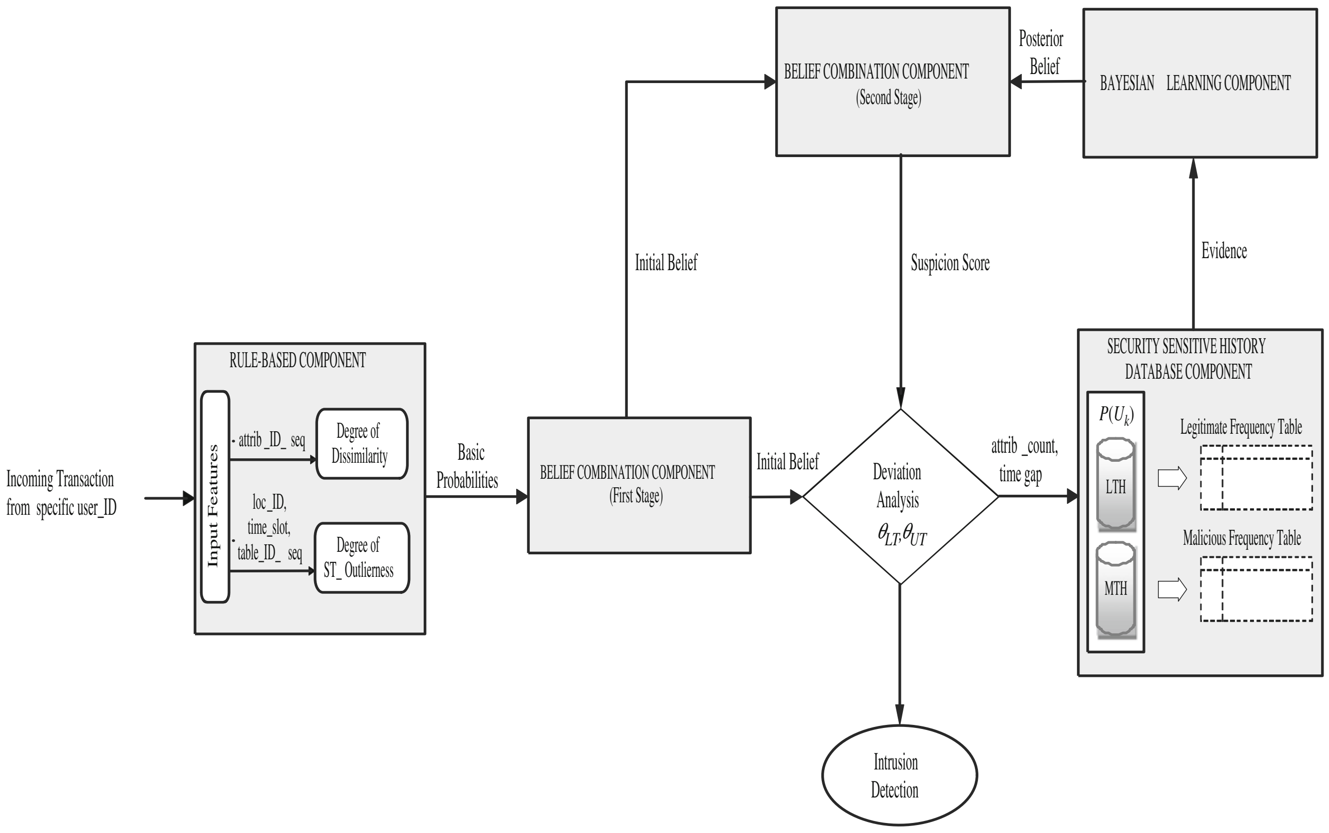
BELIEF COMBINATION COMPONENT (Second Stage)
Posterior Belief
BAYESIAN LEARNING COMPONENT
Initial Belief
Suspicion Score
Evidence
RULE-BASED COMPONENT
Input Features
attrib_ID_ seq
Degree of Dissimilarity
loc_ID, time_slot, table_ID_ seq
Degree of ST_ Outlierness
Incoming Transaction from specific user_ID
Basic Probabilities
BELIEF COMBINATION COMPONENT (First Stage)
Initial Belief
Deviation Analysis θ_{LT}, θ_{UT}
attrib _count, time gap
SECURITY SENSITIVE HISTORY DATABASE COMPONENT
$P(U_k)$
LTH
MTH
Legitimate Frequency Table
Malicious Frequency Table
Intrusion Detection

Fig. 1 Block diagram of the two-stage database intrusion detection system

- *loc_ID*: identifies the location where a transaction was carried out
- *time_slot*: time slot in which a transaction occurs. We have partitioned a day into 48 time slots, each of thirty minutes duration
- *table_ID_seq*: table access sequence in a transaction. Here, $table\_ID\_seq = < T1, T4 >$
- $attrib\_count = \{HSWC, HSRC, MSWC, MSRC, LSWC, LSRC\}$. It gives the count of the different types of attributes accessed in a transaction based on their sensitivity where:

  (a) *HSWC* (High Sensitive Write Count): number of high sensitive attributes modified in a transaction. In this Eg. $HSWC = 1$
  (b) *HSRC* (High Sensitive Read Count): number of high sensitive attributes read in a transaction. In this Eg. $HSRC = 1$
  (c) *MSWC* (Medium Sensitive Write Count): number of medium sensitive attributes modified in a transaction. In this Eg. $MSWC = 0$
  (d) *MSRC* (Medium Sensitive Read Count): number of medium sensitive attributes read in a transaction. In this Eg. $MSRC = 1$
  (e) *LSWC* (Low Sensitive Write Count): number of low sensitive attributes modified in a transaction. In this Eg. $LSWC = 0$
  (f) *LSRC* (Low Sensitive Read Count): number of low sensitive attributes read in a transaction. In this Eg. $LSRC = 2$

- ρ is the time gap from the previous transaction by the same user

3. $\psi(T_{j,\rho}^{U_k})$ is the suspicion score of the j^{th} transaction $T_{j,\rho}^{U_k}$ by user U_k
4. θ_{LT} is the lower threshold, where $\{0 \leq \theta_{\text{LT}} \leq 1\}$
5. θ_{UT} is the upper threshold, where $\{(0 \leq \theta_{\text{UT}} \leq 1) \wedge (\theta_{\text{LT}} \leq \theta_{\text{UT}})\}$

3.1 TSDIDS components

To meet the functionality as identified above, a comprehensive architecture is proposed as shown in Fig. 1, which consists of the following four major components:

- Rule-based Component (RBC)
- Belief Combination Component (BCC)

– Security Sensitive History Database Component (SSHDC)
– Bayesian Learning Component (BLC)

3.1.1 Rule-based component (RBC)

The RBC consists of a number of distinct generic as well as user-specific rules which classify an incoming transaction as malicious with a certain probability. It measures the extent to which a transaction's behavior deviates from the user's normal profile for each new transaction submitted by the user. We briefly discuss two of the rule-based techniques here.

– Sequence Alignment for Deviation Detection (R_1)
  A transaction is not an arbitrary collection of queries. Each query in a transaction is chosen appropriately to achieve a meaningful purpose. A database user even submits a group of transactions to achieve certain high level goals. The database schema and the purpose to achieve a meaningful task would make the user follow a particular sequence of database operations. Therefore, forming a sequence is an effective way of representing user profiles. Once the user profile is represented using a sequence of transactional features, sequence alignment techniques can be used to raise initial concerns about any suspicious activity. As intruders are not entirely familiar with the normal database access patterns of legitimate users, some deviation is generally seen in their database access. We use heuristic based local sequence alignment tool BLAST (Altschul et al. 1990) for comparing sequence information.
  Behavioral patterns of a user are monitored by comparing his most recent activity with his history database access patterns. Each new transaction is passed through the RBC and the new attribute sequence is aligned with each of the normal profile sequences. The degree of dissimilarity (d_s) is determined based on the similarity between the new sequence and the user's normal profile sequences. We use a simple scoring system to evaluate the degree of dissimilarity. A unit match score δ $(0 < \delta \leq 1)$ is assigned to each matched element and a unit mismatch score δ' $(0 < \delta' \leq 1)$ to each mismatched element. Let L be the length of the new sequence and M be the number of matches with the aligned good sequence. The degree of dissimilarity (d_s) is then evaluated by the following expression:

$$d_s = \begin{cases} \dfrac{\delta'(L-M) - \delta M}{L} & \text{if } \delta'(L-M) > \delta M \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

  Some of the possible transactional feature granularity levels for any database transaction include—query type, sensitive attributes, all attributes, operation on attributes and a combination of any of these. In the current work, we use *attrib_ID_seq* as the transactional feature for sequence alignment based deviation detection. The algorithm can be extended to include other transactional features as well.

– Spatio-Temporal Outlier Detection (R_2)
  In the real physical world, an individual exhibits certain spatio-temporal characteristics which signify the correlation of a person's behavior with both time and location. Similar transactions carried out by a user at certain location and time can be visualized as part of a cluster. Analysis of the spatio-temporal characteristics of a user's current behavior gives useful information on abnormal behavior in terms of his position (space) and time of accessing the database. Thus, the normal spatio-temporal profile associated with each user is mined and used for the detection of intrusive activities in databases. It may be noted that such spatio-temporal coordinates are especially meaningful in the context of mobile computing, in which users access a database through mobile devices like laptop, PDA, mobile phone, etc. If the users are predominantly static, spatio-temporal access patterns degrade to temporal access patterns.
  Since an intruder is not likely to have complete knowledge regarding the normal spatio-temporal access patterns of users, some deviation from the user's profile is usually observed in his transactions, which are detected as spatio-temporal outliers. We have applied a spatio-temporal filtering method to detect spatio-temporal outliers which are observations that are uncorrelated with the remainder of the dataset in space and time. A spatio-temporal outlier (ST-outlier) can be defined as a spatio-temporal referenced object whose thematic attribute values are significantly different from those of other spatially and temporally referenced objects in its spatial and temporal neighborhood.
  An approach based on the distance-based outlier (DB-outlier) detection technique (Knorr et al. 2000) is utilized to filter out ST-outliers. Other existing methods for outlier detection can only deal efficiently with two dimensions or attributes of a dataset. However, the concept of DB-outlier can be applied to detect outliers effectively for any dimensional dataset. Let N be the number of objects in the input dataset T and let DF be the underlying distance function that gives the distance between

any pair of objects in T. An object O in a dataset T is considered to be a $DB(p,d)$ outlier if at least a fraction p of the objects in T lie greater than a distance d from O. The parameter p is the minimum fraction of objects in a data space that must be outside an outlier's d-neighborhood denoted as d_N. For an object O, d_N of O contains the set of objects $Q \in T$ that are within distance d of O, i.e., $d_N = \{Q \in T | DF(O,Q) \leq d\}$. Let M represent the maximum number of data points within an outlier's d_N (i.e., $M = N(1-p)$). It means that an outlier needs to have less than M objects within its d_N. Thus, for object O, if $|d_N| \geq (M+1)$, then the object is considered as non-outlier. Otherwise, the point is reported to be an outlier.

Formally, let $C' = \{c_1, c_2, ..., c_n\}$ denote the clusters in a database D for a specific user and $A = \{a_1, a_2, ..., a_n\}$ be the set of attributes used to generate these clusters. The clusters can be formed by using different attributes, although in the current work, we use the attributes $\langle loc\_ID, time\_slot, table\_ID\_seq\rangle$ for generating ST-outliers. The algorithm can also be extended to include other attributes. We apply the most commonly used distance measure, specifically Euclidean distance, to compute the distance function DF which can be expressed as follows:

$$DF = \sqrt{(loc\_diff)^2 + (time\_diff)^2 + (tdist\_diff)^2} \tag{2}$$

where loc_diff: distance between current transaction location and the user's normal profile transaction location, $time\_diff$: distance between current transaction time slot and the user's normal profile time slot, $tdist\_diff$: schema distance between current transaction $table\_ID\_seq$ and the user's normal profile $table\_ID\_seq$.

For computing $tdist\_diff$, we use a distance measure similar to that suggested in Chung et al. (1999). We assume a database schema S with a set RS of relation schemas. Attributes are structurally close if they belong to the same relation or can be related by exploiting a sequence of foreign key dependencies. Consider two attributes $a_i \in r_1$, $a_j \in r_2$ where $r_1, r_2 \in RS$. The pairwise schema distance between a_i and a_j, denoted by $PS\_Dist(a_i, a_j)$ is defined as:

$$PS\_Dist(a_i, a_j) = \frac{SD(r_1, r_2)}{max\{SD(r_k, r_l) | r_k, r_l \in RS\}} \tag{3}$$

where $SD(r_1, r_2)$ is the shortest distance between r_1 and r_2 based on the primary and foreign keys by which they can be related. Given a set of attributes $A = \{a_1, a_2, ..., a_n\} \subseteq attributes(S)$, the schema distance function denoted by $tdist\_diff$, is defined as:

$$tdist\_diff(a_1, ..., a_n) = avg\{PS\_dist(a_i, a_j)\} \tag{4}$$

We measure the extent of deviation of an incoming transaction by its degree of ST_outlierness. High score indicates high possibility of being an ST-outlier. Suppose $DF_{\text{avg}}(T^{U_k}_{j,\rho})$ and $DF_{\max}(T^{U_k}_{j,\rho})$ denote average distance and maximum distance of an outlier transaction $T^{U_k}_{j,\rho}$ from the set of existing clusters in C' respectively. The degree of ST_outlierness (d_{STO}) of $T^{U_k}_{j,\rho}$ is then given by:

$$d_{\text{STO}} = \begin{cases} \dfrac{DF_{\text{avg}}(T^{U_k}_{j,\rho})}{DF_{\max}(T^{U_k}_{j,\rho})} & \text{if } |d_N| \leq M \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

We felt the necessity of the rule-based component in TSDIDS not only for the inclusion of useful features from existing systems but also to avoid handling millions of transactions which are carried out due to routine use of databases. The RBC separates out most of the easily recognizable genuine transactions from the rest.

Each of these rules R_1 and R_2 gives independent evidences that results in some beliefs about the transaction's maliciousness or genuineness. The suspicion about the transaction is more intensified by combining the evidences from the rules, which is handled by the belief combination component of TSDIDS. We have exploited two specific techniques as rules in the RBC for the current work. However, functionality of the component can be further enriched by incorporating new rules according to existing trends and for each user, the parameters used in the rule would vary depending on his pattern of access.

3.1.2 Belief combination component (BCC)

The role of the BCC is to combine evidences from the rules R_1 and R_2 and compute an initial belief for each transaction submitted to the TSDIDS. It may be noted that some attempts have been made to apply Dempster–Shafer theory (DST) to computer security. Wang et al. (2004) present a distributed intrusion detection system, which uses DST to combine evidences from distributed sensors. They show that multi-sensor data fusion scheme gives better performance than a single sensor. Chen and Venkataramanan (2005) have applied Dempster–Shafer approach to distributed intrusion detection in ad hoc networks. Data from multiple nodes are combined to estimate the likelihood of intrusion. A useful application of DST is covered in the work of Yi et al. (2000). They have introduced a novel way of using the conflict value in DST for a given sensor model and experimentally shown considerable improvement in performance. Panigrahi et al. (2007) have used DST for fraud detection in mobile communication networks.

The basic DST is a mathematical theory of evidence based on belief functions and plausible reasoning. It assumes a Universe of Discourse (UD), also called the Frame of Discernment, which is a set of mutually exclusive and exhaustive possibilities (Shafer 1976). For every incoming transaction $T_{j,\rho}^{U_k}$, the rules R_1 and R_2 contribute their independent observations about the behavior of the transaction. Dempster's rule for combination (Sentz 2002) gives a numerical procedure for combining together observations from the RBC to compute an initial belief for a transaction. Two basic probability assignments $m_1(h)$ and $m_2(h)$ are combined into a third basic probability assignment $m(h)$ by the following Dempster's combination rule:

$$m(h) = m_1(h) \oplus m_2(h) = X \sum_{x \cap y = h} m_1(x) m_2(y) \tag{6}$$

where, '$\oplus$' represents the Dempster's combination operator that combines two basic probability assignments into a third basic probability assignment and X is the normalization constant defined by the following Eq. 7:

$$X = \frac{1}{K} \tag{7}$$

$$K = 1 - \sum_{x \cap y = \phi} m_1(x) m_2(y) \tag{8}$$

However, the basic DST has some major drawbacks. It does not quite well model evidence with a higher degree of conflict. The degree of conflict refers to the lack of commonness or agreement among the evidence obtained from independent sources. The normalization constant in the Dempster's combination rule (Eq. 6) has the effect of completely ignoring conflict and consequently, this operation yields counterintuitive results in the face of significant conflict in certain contexts. Dempster's combination operator is a poor solution for the management of conflict between the various information sources. Moreover, the conflict increases with the number of information sources. That is why a strategy for re-assigning the conflicting mass is essential.

To solve this problem, we have employed the Extended Dempster–Shafer theory (EDST) proposed by Campos and Cavalcante (2003) which presents a new improved rule for combining evidences. EDST overcomes the above mentioned pitfalls, allowing the combination of evidences with higher degrees of conflict reliably and rationally. The EDST combination rule assigns the beliefs according to the degree of conflict between the evidences and assigns the remaining belief to the environment and not to the common hypothesis. It makes possible to combine evidences with most of their beliefs assigned to disjoint hypothesis without the side effect of a counterintuitive behavior. The conflict between two belief functions bel_1 and bel_2, denoted by $Con(bel_1, bel_2)$ is given by the logarithm of the normalization constant as follows:

$$Con(bel_1, bel_2) = \log(X) \tag{9}$$

If there is no conflict between bel_1 and bel_2, $Con(bel_1, bel_2) = 0$ and if there is nothing in common between the two evidences, then $Con(bel_1, bel_2) = \infty$. The modified Dempster's combination rule automatically incorporates the uncertainty coming form the conflicting evidences which is given by the following Eq. 10:

$$m(h) = m_1(h) \oplus m_2(h) = \frac{X \sum_{x \cap y = h} m_1(x) m_2(y)}{1 + \log\left(\frac{1}{K}\right)} \tag{10}$$

For the database intrusion detection problem, EDST is more relevant as compared to other fusion methods since it introduces a new rule of combination that embodies the conflict among the evidences. It provides a rule for computing the confidence measures of three states of knowledge: intrusion (I), ¬intrusion ($\neg I$) and suspicious (unknown) based on data from new as well as old evidence. Hence, we use EDST for combining evidences for this problem. The UD consists of two possible values for any suspected transaction $T_{j,\rho}^{U_k}$ which is given as $UD = \{I, \neg I\}$. For this UD, the power set has three possible elements: hypothesis $h = \{I\}$ implying that $T_{j,\rho}^{U_k}$ is intrusive, hypothesis $\bar{h} = \{\neg I\}$ that it isn't, and universe hypothesis UD that $T_{j,\rho}^{U_k}$ is suspicious. The Basic Probability Assignments (BPAs) for the two rules R_1 and R_2 can now be given as follows:

- BPA for R_1: For a transaction in which *attrib_ID_seq* does not match completely with the normal profile *attrib_ID_seq*, we make the following basic probability assignments using the degree of dissimilarity (d_s) given by Eq. 1:

$$m_1(h) = \frac{\delta'(L - M) - \delta M}{L}$$

$$m_1(\bar{h}) = 0$$

$$m_1(UD) = 1 - \left(\frac{\delta'(L - M) - \delta M}{L}\right) \tag{11}$$

– BPA for R_2: For a transaction detected as an ST-outlier, we make the following basic probability assignments using the degree of ST_outlierness (d_{STO}) given by Eq. 5:

$$m_2(h) = \frac{DF_{\text{avg}}(T^{U_k}_{j,\rho})}{DF_{\text{max}}(T^{U_k}_{j,\rho})}$$
$$m_2(\overline{h}) = 0$$
$$m_2(UD) = 1 - \left(\frac{DF_{\text{avg}}(T^{U_k}_{j,\rho})}{DF_{\text{max}}(T^{U_k}_{j,\rho})}\right) \quad (12)$$

The zero in the BPA of $\overline{h}$ in Eqs. 11 and 12 does not imply impossibility. It means that neither of the rules R_1 and R_2 gives any support to the belief that transaction $T^{U_k}_{j,\rho}$ is genuine. Following Eq. 10, the combined belief of R_1 and R_2 in h is expressed as:

$$P(h) = m_1(h) \oplus m_2(h) \quad (13)$$

Based on the initial belief $P(h)$, a transaction can be initially classified as legitimate, malicious or suspicious. Since $P(h)$ and $P(\overline{h})$ add to unity, $P(\overline{h}) = 1 - P(h)$.

3.1.3 Security sensitive history database component (SSHDC)

In many decision making situations, an intermediate possibility arises for which more information (evidence) regarding the user's database access behavior needs to be obtained prior to deciding. Once a transaction from a user is labeled as suspicious, further transactions from this particular user are permitted but each new transaction is investigated by the SSHDC component of TSDIDS.

For tracking such suspicious transactions, a large volume of history database transactions is collected and warehoused in the SSHDC. This is done to avoid troubling the legitimate users who make occasional high level of activity. Thus, we have built a legitimate transactions history (LTH) for individual users from their past behavior and a generic malicious transactions history (MTH) from different types of past intrusive data, taking into consideration the sensitivity levels of table attributes. It should be noted that when an intruder attacks through the login of a new user, there is no history for that particular user. In such a situation, the proposed model reduces to a misuse detection system, that recognizes the attacks by comparing the current activity against the known patterns of abuse.

In recent years, database size has grown considerably in terms of the number of tuples (objects) and number of attributes (fields) in the database. It is now quite common to have databases containing of the order of 10^9 tuples, each having 10^2 or 10^3 attributes (Fayyad et al. 1996). However, in every database, there are a few attributes that are more important to be tracked for malicious modifications or leakage as compared to other attributes. By grouping the attributes according to the relative order of importance based on their sensitivity, it becomes comparatively easier to track only those sensitive attributes whose modification or leakage has larger impact on the database security. It has been observed that intrusion detection systems often raise a large number of alarms, many of which are triggered incorrectly by benign events (Julisch and Dacier 2002). By classification of the attributes, the administrator needs to investigate only the alarms generated due to the unexpected modification of sensitive attributes instead of checking all the attributes. Since the goal of a DIDS is to minimize the losses suffered by the customers and organizations, it is important to track the high sensitive attributes more carefully.

We categorize the attributes into the following three sensitivity levels—High Sensitivity (HS), Medium Sensitivity (MS) and Low Sensitivity (LS). Also, modification (write) of an attribute of a particular sensitivity level is considered more important than accessing (read) the same attribute, from database integrity point of view. We consider an attribute say x, then $W(x_w) > W(x_r)$, where W is a weight function, x_w denotes writing or modifying attribute x and x_r denotes reading of attribute x.

For a given schema, we define six types of operations on the attributes based on the different sensitivity levels and mode of access. Numerical weights are assigned to each operation, which signify their relative order of importance. The six types of operations (op) are: High Sensitive Write (HSW), High Sensitive Read (HSR), Medium Sensitive Write (MSW), Medium Sensitive Read (MSR), Low Sensitive Write (LSW) and Low Sensitive Read (LSR) such that, $W_{\text{HSW}} > W_{\text{HSR}} > W_{\text{MSW}} > W_{\text{MSR}} > W_{\text{LSW}} > W_{\text{LSR}}$. The weight of a given *attrib_ID_seq* of a certain transaction is same as the weight of the most sensitive operation applied on the attributes in that sequence.

When a transaction is found to be suspicious, the initial observation done by the rule-based component is further strengthened by monitoring the frequency of the most sensitive operation in the *time_gap* (ρ) from the last transaction by the same user. For accomplishing this, we divide the *time_gap* (ρ) into four units such that, $\rho \in \{1, 2, 3, 4\}$ where $\rho = 1 \Rightarrow 0 < time\_gap \leq 8$, $\rho = 2 \Rightarrow 8 < time\_gap \leq 16$, $\rho = 3 \Rightarrow 16 < time\_gap \leq 24$ and $\rho = 4 \Rightarrow time\_gap > 24$. It may be noted that the *time_gap* values usually differ from user to user based on their access behavior. However, in the present work we have chosen fixed values for the purpose of experimentation. The experiments can be repeated by choosing any other suitable values or by clustering past data to determine user-specific time gaps.

We define 24 mutually exclusive and exhaustive events $D_{\text{op}\rho}$ by considering the four *time_gap* units (ρ) and the six types of operations (*op*) as discussed above. Occurrence of each event $D_{\text{op}\rho}$ depends on the most sensitive operation *op* carried out in a transaction, where $op \in \{HSW, HSR, MSW, MSR, LSW, LSR\}$, and the *time_gap* unit (ρ) in which a transaction occurs. The set of events is expressed as: $D_{\text{op}\rho} = \{D_{\text{HSW1}}, D_{\text{HSW2}}, D_{\text{HSW3}}, D_{\text{HSW4}}, \ldots, D_{\text{LSR3}}, D_{\text{LSR4}}\}$. The event D_{HSW1} is defined as the occurrence of a transaction $T^{U_k}_{j,\rho}$ by the same user U_k in which an HSW operation is performed at $\rho = 1$ ($0 < time\_gap \leq 8$, i.e., another transaction is carried out by the same user within 8 hours of the last transaction) which can be represented as:

$$D_{\text{HSW1}} = True|\{\exists T^{U_k}_{j,\rho} \wedge (op = HSW \wedge (\rho = 1))\} \quad (14)$$

Similarly, the events D_{HSW2}, D_{HSW3} and D_{HSW4} can be expressed as:

$$D_{\text{HSW2}} = True|\{\exists T^{U_k}_{j,\rho} \wedge (op = HSW \wedge (\rho = 2))\} \quad (15)$$

$$D_{\text{HSW3}} = True|\{\exists T^{U_k}_{j,\rho} \wedge (op = HSW \wedge (\rho = 3))\} \quad (16)$$

$$D_{\text{HSW4}} = True|\{\exists T^{U_k}_{j,\rho} \wedge (op = HSW \wedge (\rho = 4))\} \quad (17)$$

The definition of the remaining events follows from the above. It may be noted that, we chose the above definitions of $D_{\text{op}\rho}$s to handle frequent as well as infrequent users during experimentation. In the current work, we have carried out various experiments by taking a specific case, in which twenty four events have been defined with eight hours of time gap between them. However, any number of events with other values of time gap can be defined similarly. Building the user profiles at a more detailed level of granularity results in lower false positives. However, building the transaction history databases and maintaining its consistency would become more complex and rigorous. Therefore, we have defined only twenty four events for simplifying the profile building process. Other values could be similarly defined. User-specific definitions of $D_{\text{op}\rho}$s can also be derived by clustering *time_gap* for each user from the history data.

We next compute $P(D_{\text{op}\rho}|h)$ and $P(D_{\text{op}\rho}|\overline{h})$ from the MTH and the LTH respectively. $P(D_{\text{op}\rho}|h)$ measures the probability of occurrence of $D_{\text{op}\rho}$ given that a transaction is originating from an intruder and $P(D_{\text{op}\rho}|\overline{h})$ measures the probability of occurrence of $D_{\text{op}\rho}$ given that it is genuine. The likelihood functions $P(D_{\text{op}\rho}|h)$ and $P(D_{\text{op}\rho}|\overline{h})$ are given by the following expressions:

$$P(D_{\text{op}\rho}|h) = \frac{\#(\text{Occurrences of } D_{\text{op}\rho} \text{ in MTH})}{\sum_{\rho=1}^{4} \#(\text{Occurrences of } D_{\text{op}\rho} \text{ in MTH})} \quad (18)$$

$$P(D_{\text{op}\rho}|\overline{h}) = \frac{\#(\text{Occurrences of } D_{\text{op}\rho} \text{ by } U_k \text{ in LTH})}{\sum_{\rho=1}^{4} \#(\text{Occurrences of } D_{\text{op}\rho} \text{ by } U_k \text{ in LTH})} \quad (19)$$

We have created two look-up tables MFT (Malicious Frequency Table) and LFT (Legitimate Frequency Table) to maintain the values of $P(D_{\text{op}\rho}|h)$ and $P(D_{\text{op}\rho}|\overline{h})$ respectively. Using Eqs. 18 and 19, $P(D_{\text{op}\rho})$ can be computed as follows:

$$P(D_{\text{op}\rho}) = P(D_{\text{op}\rho}|h)P(h) + P(D_{\text{op}\rho}|\overline{h})P(\overline{h}) \quad (20)$$

We update the SSHDC frequently in order to retain the accuracy of TSDIDS, thus reducing the number of false alarms. SSHDC update is an offline procedure.

3.1.4 Bayesian learning component (BLC)

Bayesian learning is a tool to measure evidences supporting alternative hypotheses and arrive at optimal decisions. It gives a formal and consistent way of reasoning in presence of uncertainty. We use Bayesian learning to update the suspicion score (ψ) of a transaction after getting the new evidence $D_{\text{op}\rho}$ from the SSHDC. ψ gives the probability that the current transaction is intrusive. Belief update is done by using the Bayes rule, which is given by the following Eq. 21:

$$P(h|D_{\text{op}\rho}) = \frac{P(D_{\text{op}\rho}|h)P(h)}{P(D_{\text{op}\rho})} \quad (21)$$

By substituting Eq. 20 in Eq. 21 we get:

$$P(h|D_{\text{op}\rho}) = \frac{P(D_{\text{op}\rho}|h)P(h)}{P(D_{\text{op}\rho}|h)P(h) + P(D_{\text{op}\rho}|\overline{h})P(\overline{h})} \quad (22)$$

The goal of Bayesian learning is to find the most probable hypothesis h_{map} given the training data. This is known as the Maximum A Posteriori Hypothesis (MAP Hypothesis) which can be expressed as:

$$h_{\text{map}} = \max_{h \in H} P(h|D_{\text{op}\rho}) \quad (23)$$

Thus, for each hypothesis h in the hypothesis space H, we calculate the posterior probability $P(h|D_{\text{op}\rho})$

and $P(\overline{h}|D_{\text{op}\rho})$ by using Bayes rule and then output the hypothesis with the highest posterior probability as h_{map}. The database intrusion detection problem has the following two hypotheses, $h : I$ and $\overline{h} : \neg I$. By substituting the values obtained from Eqs. 13, 18 and 19 in Eq. 22, the posterior probability for hypothesis $h : I$ is given as:

$$P(I|D_{\text{op}\rho}) = \frac{P(D_{\text{op}\rho}|I)P(I)}{P(D_{\text{op}\rho}|I)P(I) + P(D_{\text{op}\rho}|\neg I)P(\neg I)} \tag{24}$$

Similarly, the posterior probability for hypothesis $\overline{h} : \neg I$ is given as:

$$P(\neg I|D_{\text{op}\rho}) = \frac{P(D_{\text{op}\rho}|\neg I)P(\neg I)}{P(D_{\text{op}\rho}|I)P(I) + P(D_{\text{op}\rho}|\neg I)P(\neg I)} \tag{25}$$

where I signifies intrusion. Depending on which of the two posterior values is greater, future actions are decided by the TSDIDS.

3.2 Methodology

Each incoming transaction is first examined by the rule-based component of the system. Basic probability values $BPA(R_1)$ and $BPA(R_2)$ assigned by the RBC are combined using the BCC to get the initial belief $P(h)$ for the transaction. If $P(h) < \theta_{\text{LT}}$, the transaction is considered to be genuine and is allowed to go through. On the other hand, if $P(h) > \theta_{\text{UT}}$ then the transaction is declared as malicious and manual confirmation can be made with the legitimate user. In case $\theta_{\text{LT}} \leq P(h) \leq \theta_{\text{UT}}$, the transaction is allowed but the *user_ID* corresponding to the user is labeled as suspicious. If this is the first suspicious transaction carried out by the user, then the corresponding *user_ID* is inserted into a *suspect_table*. TSDIDS then waits until the next transaction occurs using the same *user_ID*.

When the next transaction occurs for the same user, it is again passed through TSDIDS. RBC assigns basic probabilities and BCC computes the initial belief $P(h)$ for the new transaction. In case the transaction is found to be suspicious, it is once more inserted into the *suspect_table*. Since each transaction is time stamped, from the *time_gap* (ρ) between the current and previous transaction and the most sensitive operation applied on the *attrib_ID_seq* in the current transaction, our detection system determines which event E has occurred out of the twenty four $D_{\text{op}\rho}$s and retrieves the corresponding $P(E|h)$ and $P(E|\overline{h})$ values from the tables MFT and LFT, respectively. The posterior beliefs $P(h|E)$ and $P(\overline{h}|E)$ are next computed using Eqs. 24 and 25 and MAP hypothesis (Eq. 23) is applied.

$P(h|E)$ and $P(\overline{h}|E)$ are the updated beliefs about the last transaction by the user based on the evidence from SSHDC and previous round suspicion score ψ (last round). Since for the second suspicious transaction on a user, there is no ψ (last round), the $P(h)$ value of the first round is itself taken as ψ (last round) and posterior beliefs are computed based on this value. If $P(h|E) \geq P(\overline{h}|E)$, then the TSDIDS applies the extended D-S rule of combination to get the suspicion score ψ (current round) by combining $P(h|E)$ and current round $P(h)$. The current round ψ value is inserted into the suspect table at the end of each round unless the suspicion score falls below θ_{LT}. Whenever a transaction is found to be malicious and the abnormal behavior is confirmed from the user, the corresponding *user_ID* and associated transactions are moved from the LTH to the MTH in order to maintain the consistency of the SSHDC and to build the MTH.

It is to be noted that, the proposed model is able to catch the outsides as well as the malicious insiders. An outsider usually does not know what a typical user pattern is. Thus, activities of these intruders grossly deviate from normal activities and are easily detected. However, the insiders are particularly difficult to defend against as they can submit transactions similar to the genuine ones as discussed earlier in Section 1. Therefore, our proposed two-stage DIDS emphasizes on detecting the insider attacks by identifying any inconsistency or deviation of user activities from the normal profile.

3.3 An example scenario

In Table 1, we show sample results over two rounds of our proposed methodology to exemplify the system's workflow. Let us consider $\theta_{\text{LT}} = 0.3$ and $\theta_{\text{LT}} = 0.8$. Suppose initial belief $P(h) = 0.47$ which is obtained by combining the evidences from rules R_1 and R_2 by extended D-S combination rule (Eq. 10). Since $\theta_{\text{LT}} \leq 0.47 \leq \theta_{\text{UT}}$, the transaction is labeled as suspicious. We assume that it is the first suspicious transaction on this *user_ID* and hence, the transaction is entered into the *suspect_table*.

When the subsequent transaction occurs from the same user, it is again passed to the RBC and suppose we

Table 1 Sample result of TSDIDS over various rounds

Round	Initial belief	Posterior belief	Suspicion score
1	0.47	–	0.47
2	0.59	0.65	0.85

get $P(h) = 0.59$. The transaction is once more found to be suspicious. We assume that the current transaction occurs on the same *user_ID* within 8 h of the last transaction and the most sensitive operation performed on the *attrib_ID_seq* by the current transaction be HSW. From the *time_gap* ($\rho = 1$) and the most sensitive attribute operation performed by the transaction ($op = HSW$), we determine that the event D_{HSW1} has occurred and retrieve the corresponding $P(D_{\mathrm{HSW1}}|h)$ and $P(D_{\mathrm{HSW1}}|\overline{h})$ values from MFT and LFT, respectively. Let $P(D_{\mathrm{HSW1}}|h) = 0.248$ and $P(D_{\mathrm{HSW1}}|\overline{h}) = 0.118$. By applying Eqs. 24 and 25, we get $P(h|D_{\mathrm{HSW1}}) = 0.65$ and $P(\overline{h}|D_{\mathrm{HSW1}}) = 0.35$. Applying MAP hypothesis, it is observed that $P(h|D_{\mathrm{HSW1}}) \geq P(\overline{h}|D_{\mathrm{HSW1}})$. The suspicion score ($\psi$) of the current round is now computed by combining current round $P(h) = 0.59$ and the posterior belief $P(h|D_{\mathrm{HSW1}}) = 0.65$ using extended D-S combination rule (Eq. 10). We get $\psi = 0.85$ which is greater than the upper threshold θ_{UT}, and hence, the transaction is declared as malicious. An interesting observation from Table 1 is that although the user is not found to be strictly intrusive in the two transactions individually, however, due to the belief update by Bayesian learning, it is detected as an intrusion.

It may be noted that suspicion score may sometimes go up for a legitimate user. This would represent a situation in which the user carries out a number of unusual transactions. Similarly, suspicion score may also come down for an intruder occasionally, which represents a scenario in which the intruder's behavior matches exactly with the actual user. However, we will show in Section 4 that the system is robust enough to handle deviations from expected patterns to a large extent.

4 Experimental evaluation

We have carried out several experiments to show the efficacy of the proposed method. A system has been developed in MS-SQL Server 2000. We have used the standard transactional web benchmark (TPC-W) (Transaction Processing Performance Council 2002) schema for large scale simulation as suggested by the Transaction Processing Council (TPC). This benchmark provides us with a controlled database environment quite adequate for the evaluation of the proposed detection and learning system. In this schema, we have categorized all the attributes into three sensitivity levels, namely, low sensitive, medium sensitive and high sensitive. After categorizing the attributes, we define sixty different types of SQL queries using the attributes of this schema for accessing the database.

In this section, we first discuss the components of our transaction simulator. Then, we describe the choice of parameters of the TSDIDS and finally study the performance of TSDIDS.

4.1 Experimental setup

A simulation model was developed in order to evaluate the performance of the proposed database intrusion detection system since there is no benchmark data set for testing the accuracy of database intrusion detection systems. Even a detailed survey of literature has not revealed any reference to public availability of real data sets. In this domain, it is found that determining the desired distribution is an experimental art and requires extensive empirical tests to find the most effective distribution. We pursued experiments by taking several combinations of transactions generated by the simulator as shown in Fig. 2. The simulator generates synthetic transactions that represent the behavior of legitimate users as well as that of intruders.

It may be noted that Hu and Panda (2005) tested the performance of their database intrusion detection system on sets of normal as well as malicious synthetic data generated based on the average number of read operations immediately preceding a write operation and the average number of write operations in transactions. Srivastava et al. (2006) have used normal distribution with user specified mean (μ) and standard deviation (σ). It is seen that, these simulators are simplistic and they generate data only at the level of read and write operations. None of the existing synthetic data generation methods combine appropriate distributions for generating different parameters that may affect the performance of intrusion detection systems.

The above mentioned elementary simulation models do not have any control over the list of attributes that are accessed by a transaction. Moreover, they do not consider the arrival rate of transactions which is an important parameter for evaluating an intrusion detection system. As the transaction arrival rate from attacker increases, the probability of getting consecutive intrusive transactions also increases, which in turn increases intrusion detection probability. Further, any real-world database application normally contains malicious transactions interspersed with regular genuine transactions and these two types of transactions are generated by two different parties, namely, genuine users and intruders. Hence, these are independent events with separate arrival rates.

Moreover, database transactions are usually skewed—occurrence of intrusive transactions is very

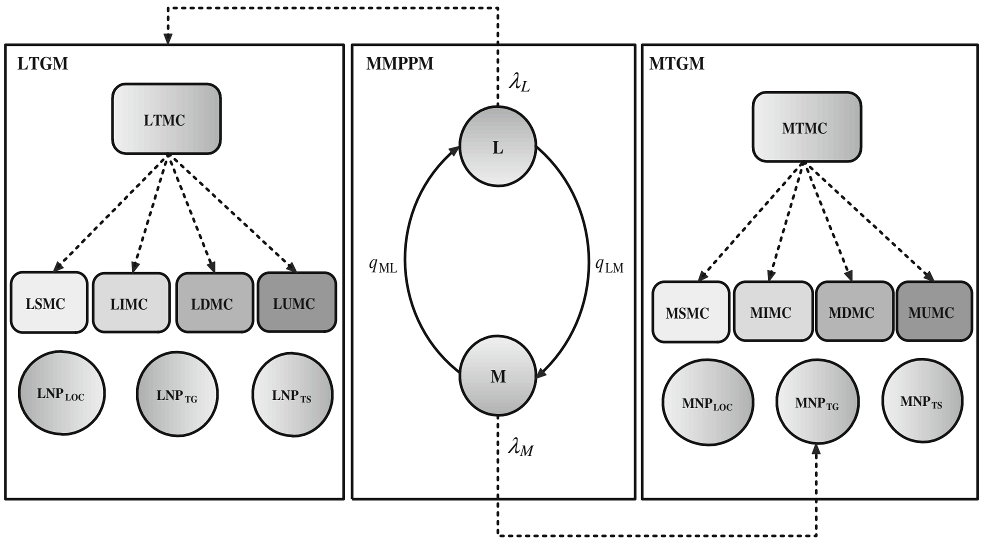

Fig. 2 Transaction simulator

low compared to genuine transactions. Hence, mixing of genuine and intrusive transactions need to be controlled properly. However, the existing synthetic data generation models are not able to control the mixing of genuine and intrusive transactions as well as the arrival rate. Therefore, we have constructed a comprehensive transaction simulator using a Markov Modulated Poisson Process (MMPP) (A Discrete Time Markov Chain (DTMC)) whose states determine the arrival rates of the Poisson Processes. The MMPP can control mixing of transactions along with the transaction arrival rate from attacker and normal user. Besides, three Gaussian distribution functions are used for handling various user profiles and different categories of intruders (based on *loc_ID*, *time_gap* and *time_slot*). This simulator additionally supports hierarchical level of Markov Chains for monitoring the formation of transactions using four basic types of queries—select, insert, delete and update. Transaction generation is regulated in our simulator at the level of table attribute, query and type of query.

Before describing our experimental findings, we give a brief outline of our simulator components as shown in Fig. 2 as well as the generation procedure for legitimate transactions, malicious transactions and their appropriate mixing.

– Markov Modulated Poisson Process Module (MMPPM)
  A *Markov Modulated Poisson Process* is a doubly stochastic Poisson process where the default rate is determined by the state of the underlying continuous-time Markov chain. Since the Markov chain has a finite number of states, the Poisson arrval rate (λ) takes discrete values corresponding to each state. λ is expressed as the average number of arrivals during a unit period of time. The central idea is to model the behavior of genuine users as well as intruders through an Markov Modulated Poisson Process (MMPP). The proposed MMPPM uses a 2-state MMPP consisting of a legitimate state L and a malicious state M with arrival rates λ_L and λ_M respectively. Mixing of legitimate and malicious transactions is controlled by the L and M states of the MMPPM. Transition from L to M takes place with probability q_{LM} and from M to L with probability q_{ML}. It may be noted that, in the intrusion detection domain, the occurrence of malicious transactions is very sparse as compared to the genuine transactions (Axelsson 2000). For handling various such real life scenarios, we vary different simulation parameters like λ_L, λ_M, q_{LM} and q_{ML} that affect the overall working of the system.
– Legitimate Transaction Generation Module (LTGM)
  LTGM is used to generate synthetic legitimate transactions comprising of four basic types of queries—select, insert, delete and update. This module consists of five finite Markov chains—Legitimate Select Markov Chain (LSMC), Legitimate Insert Markov Chain (LIMC), Legitimate Delete Markov Chain (LDMC), Legitimate Update Markov Chain (LUMC) and Legitimate Transaction Markov Chain (LTMC). Each Markov chain has an associated transition probability matrix (TPM) and an initial probability distribution vector (IPDV). If the number of select type queries be N, then the number of states in the LSMC is

also N. The Markov chains LIMC, LDMC and LUMC are defined similarly. Each transaction is a collection of multiple queries and the formation of transactions is controlled by a Markov chain denoted as LTMC, which has four states, same as the number of query types. The number of queries in a transaction is chosen randomly within a specific lower and upper bound. Furthermore, the component LTGM also consists of three Gaussian processes with user-specified mean (μ) and standard deviation (σ)—Legitimate Gaussian Process LGP_{LOC} $\langle\mu_{\text{LLOC}}, \sigma_{\text{LLOC}}\rangle$ for generating *loc_ID*, LGP_{TG} $\langle\mu_{\text{LTG}}, \sigma_{\text{LTG}}\rangle$ for generating *time_gap* and LGP_{TS} $\langle\mu_{\text{LTS}}, \sigma_{\text{LTS}}\rangle$ for generating *time_slot* of legitimate users, as shown in Fig. 2. We use Gaussian distribution since it is the most commonly observed probability distribution in many natural processes. The TPMs and IPDVs related to the various Markov chains and the mean and standard deviation of the Gaussian processes are subjected to change during generation of normal transactions for handling different user profiles. In our experiments, we set $q_{\text{LM}} = 0$ to restrict transaction generation within the L state of the MMPPM.

– Malicious Transaction Generation Module (MTGM)
  This component is used to generate synthetic malicious transactions and is similar to LTGM. It consists of five finite Markov chains—Malicious Select Markov Chain (MSMC), Malicious Insert Markov Chain (MIMC), Malicious Delete Markov Chain (MDMC), Malicious Update Markov Chain (MUMC) and Malicious Transaction Markov Chain (MTMC). The three Gaussian processes for building MTGM are—Malicious Gaussian Process MGP_{LOC} $\langle\mu_{\text{MLOC}}, \sigma_{\text{MLOC}}\rangle$, MGP_{TG} $\langle\mu_{\text{MTG}}, \sigma_{\text{MTG}}\rangle$ and MGP_{TS} $\langle\mu_{\text{MTS}}, \sigma_{\text{MTS}}\rangle$, which are employed for generating *loc_ID*, *time_gap* and *time_slot* for intruders. We generate the malicious transactions keeping in mind the insider as well as outsider threat scenarios. In case of insider threat, anomalous query set will have high resemblance with the genuine query set. However, there may be some inter-transactional variations which is handled by changing the TPMs and IPDVs related to the various Markov chains. For outsider threat, variation is mostly seen in the intra-transactional patterns. During experimentation, we set $q_{\text{ML}} = 0$ to restrict transaction generation within the state M of the MMPPM. The mean and standard deviation of the Gaussian processes are changed during generation of malicious transactions for handling different categories of intruders.

4.2 Choice of design parameters

The effectiveness of the proposed database intrusion detection model relies on several parameters. In order to determine the impact of these parameters on our detection system, eight different simulator settings (SS1 to SS8) are chosen by varying the simulation parameters λ_{L}, λ_{M}, q_{LM}, q_{ML}, μ_{LTG} and μ_{MTG} as shown in Table 2. This achieves arrival rate variations for genuine users as well as intruders. It is seen that, occurrence of intrusive transactions reduces from SS1 to SS8 as q_{LM} and λ_{M} are decreased and μ_{MTG} is gradually increased over the settings. Proper mixing of spatio-temporal parameters μ_{LLOC}, μ_{MLOC}, μ_{LTS} and μ_{MTS} is done by choosing six diverse spatio-temporal combinations (ST1–ST6) as shown in Table 3 to capture the spatio-temporal access patterns of users.

Standard metrics are used to study the performance of the system under different test cases. True negative (TN) is the percentage of genuine transactions labeled as genuine, true positive (TP) is the percentage of intrusive transactions caught by the system (also called hit), false negative (FN) is the percentage of intrusive transactions labeled as genuine (also called miss) and false positive (FP) is the percentage of genuine transactions labeled as intrusive (also called false alarm).

Before testing the performance of the proposed system, we first perform a set of experiments to determine a good combination of the design parameters, namely, lower threshold (θ_{LT}) and upper threshold (θ_{UT}). From the discussions in Section 3, it is obvious that the effectiveness of the proposed system is dependent on the two parameters θ_{LT} and θ_{UT}. The performance of the system will degrade if these parameters are set incorrectly. If θ_{UT} is set too high, then most of the intrusions will go undetected whereas if θ_{UT} is set too low then there will be a large number of false alarms which will lead to serious denial-of-service. Similarly, high value of θ_{LT} will let most of the intrusions go through and low value of θ_{LT} will lead to unnecessary investigation of a large number of genuine transactions.

Table 2 Simulator settings for arrival rate variations

Simulator setting	q_{LM}	q_{ML}	λ_{L}	λ_{M}	μ_{LTG}	μ_{MTG}
SS1	0.50	0.50	1	4	5	1
SS2	0.15	0.50	1	4	4	1
SS3	0.15	0.70	1	4	4	2
SS4	0.10	0.80	1	4	3	2
SS5	0.10	0.80	1	2	3	3
SS6	0.10	0.90	3	1	2	4
SS7	0.05	0.96	4	1	1	5
SS8	0.05	0.99	8	1	1	5

Table 3 Simulator settings for spatio-temporal data generation

Spatio-temporal setting	μ_{LLOC}	μ_{MLOC}	μ_{LTS}	μ_{MTS}
ST1	1	2	22	2
ST2	1	3	28	4
ST3	2	3	35	12
ST4	1	4	24	34
ST5	2	1	36	39
ST6	1	5	30	41

Hence, selection of θ_{LT} and θ_{UT} has an associated trade-off. We, therefore, carried out experiments to determine a good choice of the θ_{LT}, θ_{UT} combination.

In Table 4, we show the variation of Mean TP/Mean FP for different values of θ_{LT} and θ_{UT}. The values shown in this table represent average of the results obtained for the eight simulator settings of Table 2 and six spatio-temporal settings of Table 3. In particular, for every combination of simulator setting $\{SSi|1 \leq i \leq 8\}$ and spatio-temporal setting $\{STj|1 \leq j \leq 6\}$, the average is estimated over 50 independent runs of the simulator consisting of 100 transactions each. Amount of space required is mainly dependent on the size of history databases LTH and MTH. In our implementation, we have used 12 features as discussed earlier in Section 3 for representing a transaction. Each transaction requires approximately 706 bytes. We have experimented with size of LTH = 1000 (user-specific) and size of MTH = 500 (generic) and considered 100 users to be present in an organization. Under these conditions, the space requirement can be determined as follows:

Space requirement = size of LTH + size of MTH = (no. of users $\times$ no. of txns for each user $\times$ size of each txn) + (no. of txns $\times$ size of each txn) = (100 $\times$ 1,000 $\times$ 706) + (706 $\times$ 500) = 70,600,000 + 353,500 = 70,953,500 bytes $\approx$ 71 MB.

From Table 4, it is seen that as θ_{LT} increases, TP decreases reaching 79% for $\theta_{LT} = 0.35$. The same trend is true for θ_{UT} also. TP falls to 78% for $\theta_{UT} = 0.85$. FPs also show a similar trend. However, with $\theta_{LT} = 0.3$ and $\theta_{UT} = 0.7$, the difference between TP and FP is the highest. We make this as our choice since it gives a balance between the number of true positives and false positives. Thus, our design parameter setting is $\theta_{LT} = 0.3$ and $\theta_{UT} = 0.7$, which is kept fixed for the rest of the experiments. The effectiveness of TSDIDS is also dependent on the two parameters, p and d. As discussed in Section 3, following the heuristic given by Knorr et al. (2000), we set the parameter $p = 0.9$ and $d = 8$.

Table 4 Variation of mean TP/Mean FP (%) with θ_{LT} and θ_{UT}

θ_{UT}	θ_{LT}			
	0.2	0.25	0.3	0.35
0.7	84.5/9.2	83.4/8.7	*83.2/5.1*	79/5
0.75	82/8.5	81.2/7.5	78.6/5	77.5/4.8
0.8	81.4/7	80/5.4	78/4.7	74/3.8
0.85	78/5	77.1/4	76.3/3.5	73.6/2.3

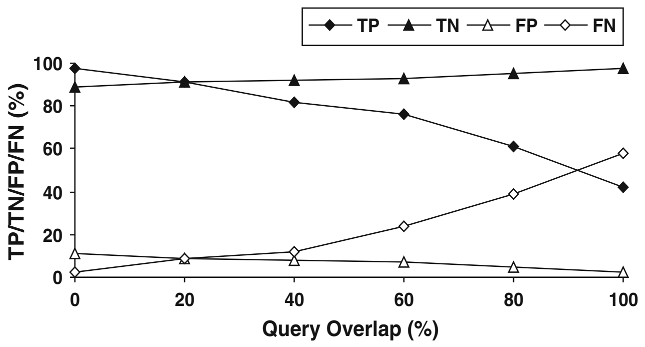

Fig. 3 Variation of TP/FP with the percentage of overlap between the malicious query set and the legitimate query set

4.3 Performance analysis

First we study the performance of the proposed database intrusion detection system with respect to changes in the percentage of overlap between the malicious query set and the legitimate query set. As discussed earlier in Section 1, insiders are potentially familiar with the organizational day-to-day activities and can submit transactions similar, though not exactly the same, to the genuine ones. Thus, the percentage of overlap between the malicious query set and the legitimate query set will be generally much higher for transactions generated by a malicious insider as compared to those launched by outsiders.

It is evident from the plot shown in Fig. 3 that the true positive rate gradually decreases (or false negative rate increases at the same rate) with increase in the percentage of overlapping queries. The figure shows that even at the point of 40% overlap, our system gives 76% TP, 93% TN, 7% FP and 24% FN. However,

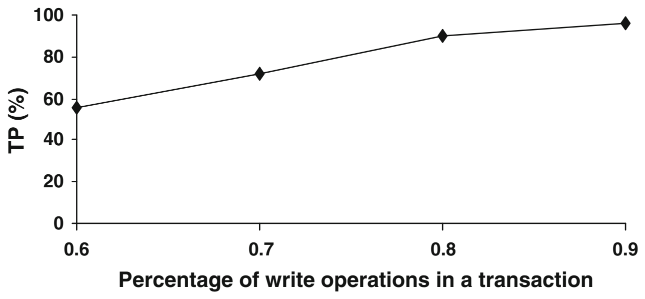

Fig. 4 Variation of TP with the percentage of write operations in a transaction

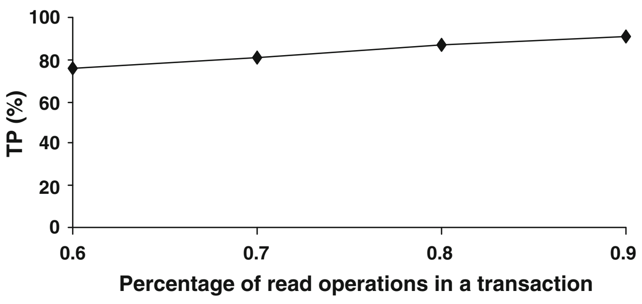

Fig. 5 Variation of TP with the percentage of read operations in a transaction

the performance is worst (lowest TP) at the point of complete overlap (i.e. intrusive query set and genuine query set are the same). The reason is that, with increase in the percentage of overlap, similarity among malicious query set and genuine query set increases which makes it difficult to distinguish between them leading to degraded performance. It is seen that FP rate also reduces (or true negative rate increases in the same way) with rise in the percentage of overlapping queries, but the reduction is slower.

We next study the effect of the percentage of write (insert/update) operations on the performance of the intrusion detection systems. It is seen from Fig. 4 that when the percentage of write operations increases, the detection rate increases since write operation is more important than read operation from information warfare point of view. We also examine the influence of read operations in a transaction on the accuracy of intrusion detection. The experimental results are shown in Fig. 5 which depicts that, with the increase in the percentage of read operations, the percentage of detected malicious transactions also increases. As we compare this result with the previous observation shown in Fig. 4, it is seen that the detection rate is more sensitive to the percentage of write operations in a transaction.

Finally, in Fig. 6, we show the performance of the proposed system over various rounds by plotting the

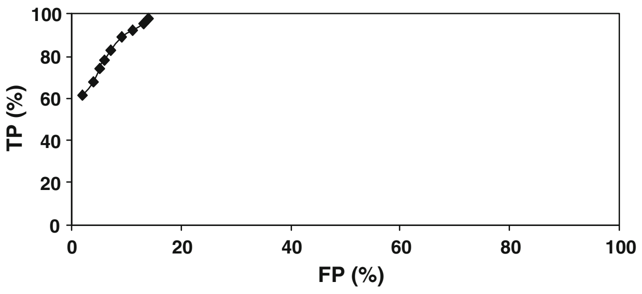

Fig. 6 Variation of TP and FP over successive rounds by ROC curve

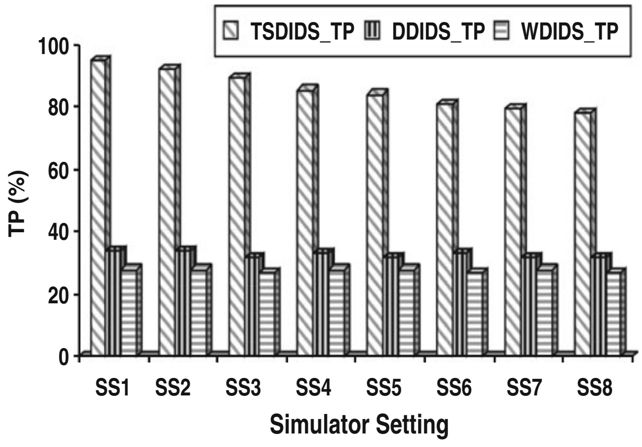

Fig. 7 Variation of TP with different simulator settings for TSDIDS, DDIDS and WDIDS

true positive (TP) and false positive (FP) in the form of Receiver Operating Characteristics (ROC) curve (Fawcett 2006). The first round commences with the first suspect transaction of a particular user. TSDIDS is able to update the belief values over successive rounds and the process continues as long as the suspicion score is within the two threshold limits. It is seen that with each successive round, the detection rate as well as the false alarm rate goes up. In the example given in Section 3.3, TSDIDS tracks the transactions of the suspicious user and the belief is updated at each round. Finally, the user was caught at the end of the second round.

It is to be noted that, the proposed model is able to catch the intrusive activities after one or more rounds depending on the extent of deviation from the user's profile. It is seen from the Fig. 6 that the detection rate gradually increases as the suspicion score increases with each round.

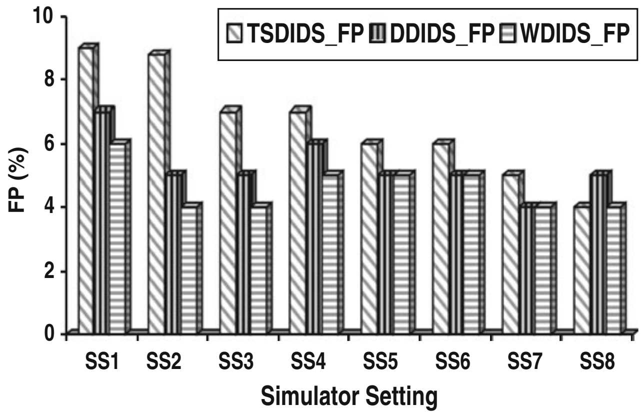

Fig. 8 Variation of FP with different simulator settings for TSDIDS, DDIDS and WDIDS

4.4 Comparative performance

We next compare the performance of our proposed database intrusion detection system with two other systems proposed respectively by Hu and Panda (2005), which uses data dependency relationships, and Srivastava et al. (2006), which uses weighted sequence mining. We use the notation DDIDS to represent the DIDS in Hu and Panda (2005) and WDIDS to represent the DIDS in Srivastava et al. (2006). Our simulator is used to generate transactions for comparative analysis, and for this set of results, the spatio-temporal parameters are set as: $\mu_{\mathrm{LLOC}} = 1$, $\mu_{\mathrm{MLOC}} = 4$, $\mu_{\mathrm{LTS}} = 24$ and $\mu_{\mathrm{MTS}} = 34$. We compute TP and FP at each SSi, $i = \{1, \ldots, 8\}$ for all the three DIDSs as mentioned above. Figure 7 shows that TSDIDS is able to detect intrusive transactions more correctly (higher TP) as compared to DDIDS and WDIDS.

It is found that choosing support and confidence values is a problem in DDIDS as well as in WDIDS. The TP rates in these two systems are strongly dependent on the number of attribute dependency rules mined. Even if a low support value is chosen, the number of rules mined is quite low, which results in degraded performance for these systems. However, the performance of TSDIDS is slightly worse for FP rate (higher value of FP) compared to both DDIDS and WDIDS as shown in Fig. 8.

5 Conclusions

Providing convenience to users by letting them access sensitive organizational data from anywhere makes it equally important to safeguard the database from intruders within or outside the organization. In this paper, a novel two-stage database intrusion detection system has been proposed which applies anomaly detection for first level inferences followed by misuse detection in the second stage. The proposed system utilizes intra-transactional as well as inter-transactional techniques for intrusion detection. A number of rules are used to analyze the deviation of an incoming transaction from the normal profile of a user. In the current work, rules like sequence alignment and spatio-temporal outlier detection are employed for measuring the extent of deviation of each new transaction submitted by a user. An extension of Dempster–Shafer theory is applied to combine multiple evidences from the rules for computation of an initial belief about each incoming transaction. A transaction is classified as normal, abnormal or suspicious depending on its initial belief. To keep track of suspicious transactions, a legitimate transactions history is built for individual users from their past behavior and a generic malicious transactions history is built from different types of past intrusive data. Moreover, three different sensitivity levels of table attributes have been considered while building the history databases to minimize the overall loss suffered by the database owner due to intrusion. For a suspicious transaction, its suspicion score is updated applying Bayesian learning based on the evidence obtained from the history databases. A final decision is made about the transaction according to its suspicion score.

Experiments were carried out on a large collection of simulated data to analyze the performance of the proposed system. The simulation yielded up to 98% TP and less than 10% FP. Use of Dempster–Shafer theory for combining rules gives good performance, especially in terms of true positives. Bayesian learning helps to further reduce the number of false alarms, which is one of the core problems of existing database intrusion detection systems.

Acknowledgements This work is partially supported by a research grant from the Department of Information Technology, Ministry of Communication and Information Technology, Government of India, under Grant No. 12(34)/04-IRSD dated 07/12/2004.

References

Altschul, S. F., Gish, W., Miller, W., Myers, W., & Lipman, J. (1990). Basic local alignment search tool. *Journal of Molecular Biology, 215*, 403–410.

Axelsson, S. (2000). The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC), 3*, 186–205.

Barbara, D., Goel, R., & Jajodia, S. (2002). Mining malicious data corruption with hidden markov models. In *Proc. 16th annual IFIP WG 11.3 working conf. on data and application security* (pp. 175–189).

Campos, F., & Cavalcante, S. (2003). An extended approach for Dempster–Shafer theory. In *Proc. IEEE int. conf. on information reuse and integration* (pp. 338–344).

Chen, T. M., & Venkataramanan, V. (2005). Dempster–Shafer theory for intrusion detection in ad hoc networks. In *Proc. IEEE internet computing* (pp. 35–41).

Chung, C. Y., Gertz, M., & Levitt, K. (1999). DEMIDS: A misuse detection system for database systems. In *Proc. integrity and internal control in information system* (pp. 159–178).

Damiani, E., Vimercati, S. D. C., Jajodia, S., Paraboschi, S., & Samarati, P. (2003). Balancing confidentiality and efficiency in untrusted relational DBMSs. In *Proc. 10th ACM conf. on computer and communications security* (pp. 93–102).

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*, 861–874.

Fayyad, U., Shapiro, G. P., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM, 39*, 27–34.

Furnell, S. (2004). Enemies within: The problem of insider attacks. *Journal of Computer Fraud & Security, 2004*(7), 6–11.

Giacinto, G., Perdisci, R., Rio, M. D., & Roli, F. (2008). Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Information Fusion, 9*, 69–82.

Goan, T. (1999). A cop on the beat: Collecting and appraising intrusion evidence. *Communications of the ACM, 42*, 46–52.

Gordon, L. A., Loeb, M. P., Lucyshyn, W., & Richardson, R. (2009). *2005 CSI/FBI computer crime and security survey*. http://www.cpppe.umd.edu/Bookstore/Documents/2005CSISurvey.pdf.

Hoglund, A. J., Hatonen, K., & Sorvari, A. S. (2000). A computer host-based user anomaly detection system using the self-organizing map. In *Proc. IEEE-INNS-ENNS int. joint conf. on neural networks (IJCNN)* (Vol. 5, pp. 411–416).

Hu, W., Hu, W., & Maybank, S. (2008). AdaBoost-based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 38*, 577–583.

Hu, Y., & Panda, B. (2005). Design and analysis of techniques for detection of malicious activities in database systems. *Journal of Network and Systems Management, 13*, 269–291.

Julisch, K., & Dacier, M. (2002). Mining intrusion detection alarms for actionable knowledge. In *Proc. ACM SIGKDD conf. on knowledge discovery and data mining* (pp. 366–375).

Kamra, A., Terzi, E., & Bertino, E. (2007). Detecting anomalous access patterns in relational databases. *The VLDB Journal, 17*, 1063–1077.

Knorr, E. M., Ng, R. T., & Tucakov, V. (2000). Distance-based outliers: Algorithms and applications. *The VLDB Journal, 8*, 237–253.

Lee, S. Y., Low, W. L., & Wong, P. Y. (2002). Learning fingerprints for a database intrusion detection system. In *Proc. 7th European symposium on research in computer security, 2502/2002* (pp. 264–280).

Lee, V., Stankovic, J., & Son, S. (2000). Intrusion detection in realtime databases via time signatures. In *Proc. 6th IEEE real-time technology and applications symposium (RTAS)* (pp. 124–133).

Lunt, T. (1996). Inside risks: Securing the information infrastructure. *Communications of the ACM, 39*, 130.

Murray, A. C. (2005). *The threat from within, network computing*. http://www.networkcomputing.com/showArticle.jhtml?articleID=166400792.

Panigrahi, S., Kundu, A., Sural, S., & Majumdar, A. K. (2007). Use of Dempster–Shafer theory and Bayesian inferencing for fraud detection in mobile communication networks. In *Proc. Australasian conf. on information security and privacy (ACISP). Lecture notes in computer science* (Vol. 4586/2007, pp. 446–460).

Panigrahi, S., Sural, S., & Majumdar, A. K. (2009). Detection of intrusive activity in databases by combining multiple evidences and belief update. In *IEEE symposium on computational intelligence in cyber security (CICS 2009)* (pp. 83–90). Nashville, Tennessee, USA.

Richardson, R. (2009). *2007 CSI computer crime and security survey*. http://i.cmpnet.com/v2.gocsi.com/pdf/CSISurvey2007.pdf.

Sentz, K. (2002). *Combination of evidence in Dempster–Shafer theory*. Sandia National Laboratories, US Department of Energy. http://www.sandia.gov/epistemic/Reports/SAND2002-0835.pdf.

Shafer, G. (1976). *A mathematical theory of evidence*. Princeton: Princeton University Press.

Srivastava, A., Sural, S., & Majumdar, A. K. (2006). Weighted intratransactional rule mining for database intrusion detection. In *Proc. Pacific-Asia knowledge discovery and data mining (PAKDD). Lecture notes in artificial intelligence, 3918/2006* (pp. 611–620). Springer.

Transaction Processing Performance Council (2002). *TPC Benchmark™ W (web commerce), specification, version 1.8*. http://www.tpc.org/tpcw/default.asp.

Triantafyllopoulos, K., & Pikoulas, J. (2002). Multivariate bayesian regression applied to the problem of network security. *Journal of Forecasting, 21*, 579–594.

Wang, Y., Yang, H., Wang, X., & Zhang, R. (2004). Distributed intrusion detection system based on data fusion method. In *Proc. 5th world congress on intelligent control and automation* (pp. 4331–4334).

Wenhui, S., & Tan, T. (2001). A novel intrusion detection system model for securing web-based database systems. In *Proc. 25th annual int. computer software and applications conf. (COMPSAC)* (pp. 249–254).

Yi, Z., Khing, H. Y., Seng, C. C., & Wei, Z. X. (2000). Multi-ultrasonic sensor fusion for mobile robots. In *Proc. IEEE intelligent vehicles symposium* (pp. 387–391).

Zhong, Y., & Qin, X. (2004). Database intrusion detection based on user query frequent itemsets mining with item constraints. In *Proc. 3rd int. conf. on information security* (pp. 224–225).

Suvasini Panigrahi is an assistant professor at the School of Computer Engineering, KIIT University, India. She received the B.Tech and M.Tech degrees in Computer Science and Engineering and Computer Science from Utkal University, India in 2002 and 2004, respectively. She received the Ph.D. degree from IIT Kharagpur in 2009. She has published various research papers on database security in refereed journals and conference proceedings. Her research interests include database systems and database security.

Shamik Sural is an associate professor at the School of Information Technology, IIT Kharagpur, India. He received the Ph.D. degree from Jadavpur University in 2000. Before joining IIT, he held technical and managerial positions in a number of organizations both in India as well as in the USA. Dr. Sural has served on the program committee of many international conferences. He is a senior member of the IEEE and a recipient of the Alexander von Humboldt Foundation Research Fellowship. He has published more than one hundred research papers in reputed international journals and conferences. His research interests include database security, data mining and multimedia database systems.

Arun K. Majumdar is a Professor of the Computer Science and Engineering Department of the Indian Institute of Technology, Kharagpur, West Bengal. He received M.Tech. and Ph.D. degrees from the University of Calcutta, in applied physics in 1968 and 1973, respectively. He also earned a Ph.D. degree in Electrical Engineering from the University of Florida, Gainesville, Florida, USA, in 1976. Professor Majumdar is currently the Deputy Director of IIT Kharagpur and earlier held other administrative positions that include Dean (Faculty and Planning), Head of the Computer Science and Engineering Department, Head of the Computer and Informatics Centre and Head of the School of Medical Science and Technology. He has been a member of several technical committees advising different

Ministries of the Central and State Governments as well as Industries and Institutions in India on information technology related matters. Before joining the IIT Kharagpur, in 1980, Professor Majumdar served as a faculty member at Indian Statistical Institute, Calcutta, and Jawaharlal Nehru University, New Delhi. He was a Visiting Professor in the Computing and Information Sciences Department of the University of Guelph, Canada in 1986–1987 and at Center for Secure Information Systems, George Mason University, Fairfax, Virginia, USA in 1999, 2003 and 2007. Professor Majumdar has more than 180 research publications in international journals and conferences. Professor Majumdar is a Fellow the Indian National Academy of Engineering, Fellow of the Institution of Engineers (India), and a Senior Member of the IEEE (USA). His research interests include Data and Knowledge based systems, Multimedia Systems, Medical Information Systems, and Information Security.