

نموذج رقم (1)

إقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Enhancement Intrusion Detection System Using Improved Naive Bayes and One-Class Classification Algorithms

تحسين أنظمة كشف الإختراقات باستخدام خوارزمية مطورة من البيز البسيطة وخوارزميات تصنيف الفئة الواحدة.

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وإن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

DECLARATION

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification

Student's name: Motaz F. Murtaja

Signature: 

Date: 3.3.2015

اسم الطالب: معترف فرج الله مرتجي

التوقيع: 

التاريخ: 3.3.2015

Islamic University of Gaza
Research and Postgraduate Affairs
Faculty of Engineering
Computer Engineering Department



الجامعة الإسلامية - غزة
شؤون الدراسات العليا والبحث العلمي
كلية الهندسة
قسم هندسة الحاسوب

**Enhancement Intrusion Detection System Using Improved Naive
Bayes and One-Class Classification Algorithms**

تحسين أنظمة كشف الإختراقات باستخدام خوارزمية مطورة من البيز البسيطة وخوارزميات
تصنيف الفئة الواحدة

By

Motaz F. Murtaja

120100821

Supervisor:

Dr. Wesam M. Ashour

**A Thesis Submitted in Partial Fulfillment of the Requirements for
the Degree of Master in Computer Engineering**

(1436هـ - 2015م)



مكتب نائب الرئيس للبحث العلمي والدراسات العليا هاتف داخلي 1150

الرقم...ج.ب.ب.ع. /35/ Ref

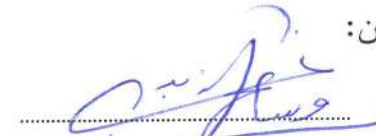


التاريخ 2015/02/22 Date

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ معترف فرج الله حمدي مرتجي لنيل درجة الماجستير في كلية الهندسة قسم هندسة الحاسوب وموضوعها:

Enhancement Intrusion Detection System Using Improved Naive Bayes and One-Class Classification Algorithms

وبعد المناقشة التي تمت اليوم الأحد 03 جمادى الأولى 1436هـ، الموافق 2015/02/22م الساعة الحادية عشرة صباحاً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:


	مشرفاً ورئيساً	د. وسام محمود عاشور
	مناقشاً داخلياً	د. محمد أحمد الحنجوري
	مناقشاً خارجياً	د. إيهاب صلاح زقوت

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في كلية الهندسة / قسم هندسة الحاسوب.

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق،،،

مساعد نائب الرئيس للبحث العلمي والدراسات العليا


أ.د. فؤاد علي العاجز



Dedication

I would like to dedicate this study

To my loving parents who have a strong and steadfast support in my journey;

To my sincere wife who give me a continuous support and understanding;

To my little daughter wishing her a great future;

To my brothers who helps me a lot;

To Mr. Yousif Alabbasi and his family for their support;

To my wonderful supervisor he was my inspiration for doing my project;

And to all interested researchers.

Acknowledgements

First and foremost, all thanks are to ALLAH the almighty, who guide me to accomplish this work, so all praise is to ALLAH. Then, there are a number of people to whom I am greatly indebted as without them this thesis might not have been written.

I would like to express my sense of gratitude and thanks to my supervisor Associate Professor Wesam Ashour Barbakh for his guidance, valuable advices, and encouragement. It was a great opportunity for me to be supervised by him. I wish also to thank the discussion Committee for their efforts.

I also wish to acknowledge my lecturer and academic staff in computer engineering department in The Islamic University of Gaza.

A special thanks to my work colleagues in Ranteesy Pediatric Hospital.

Finally, I would like to extend a very special thanks my sister for her help in linguistic review.

تقديم

يشكل الإزدياد المطرد لأنشطة اختراقات الشبكات تهديداً خطيراً على استمرارية وثبات خدمات الشبكات؛ فالشركات والأفراد يعانون من هذه الإختراقات والتدخلات الخبيثة. كذلك الشبكات وخدمات الويب بمختلف خدماتها مهددة من قبل هذه الهجمات الخبيثة على سبيل المثال : التسوق الإلكتروني، البريد الإلكتروني، الخدمات المصرفية الحكومية والخدمات الإلكترونية... الخ.

تحاول أنظمة كشف الإختراقات ملئ الفراغ في الهيكل الأمني؛ حيث إن العديد من التدخلات والإختراقات لا يزال خفياً عن تقنيات الحماية والأمن الأخرى.

العديد من الدراسات تحاول إيجاد أفضل نموذج لأنظمة كشف الإختراقات بحيث يحقق أفضل معدل كشف وأدنى معدل إنذار كاذب. فمختلف تقنيات تعليم الآلة وخوارزميات الذكاء الإصطناعي قد استخدمت في هذا المجال: شعاع الدعم الآلي، الشبكات العصبية و البيز البسيطة و خوارزميات التجميع، .. الخ.

تتقسم أنظمة كشف الإختراقات إلى نوعين رئيسيين: أنظمة الكشف المعتمدة على شيفرات البيانات وأنظمة كشف الشواذ المختلفة عن القاعدة السليمة؛ فيعتمد النوع الأول على مقارنة البيانات بأنماط البيانات المعروفة وهي تستخدم للكشف عن الهجمات معروفة الأنواع؛ أما النوع الثاني فيعتمد على كشف الهجمات عندما تحيد عن سلوك سير البيانات الطبيعي والإعتيادي.

وتقترح هذه الدراسة ثلاثة نماذج من أنظمة كشف الإختراقات: النموذج الأول يوظف خوارزمية تصنيف الفئة الواحدة كنظام لكشف الشذوذ؛ هذه الخوارزمية قائمة على توليد البيانات الصناعية من توزيع معين ليكون بيانات ذات تصنيف ذا فئتين. هذا التصنيف يمزج بين دالة الكثافة المقدرة مع احتمال الفئة المقدّر ليشكل القاعدة العامة للتصنيف؛ هذه الخوارزمية يمكن أن تتكيف مع أي أسلوب من أساليب التصنيف المتعددة لإيجاد حلول لبيانات الفئة الواحدة.

أما النموذج الثاني فهو نموذج هجين يجمع بين أنظمة الكشف المعتمدة على شيفرات البيانات وأنظمة كشف الشواذ المختلفة عن القاعدة السليمة، حيث يستخدم خوارزمية البيز البسيطة المخفية كنظام كشف معتمد على شيفرات البيانات ويستخدم خوارزميات تصنيف الفئة الواحدة كنظام لكشف الشذوذ في النموذج الهجين. يستخدم كلا من نظام تصنيف الفئة الواحدة المستخدم من قبل في النموذج الأول؛ وكذلك نظام تصنيف الفئة الواحدة بشعاع الدعم الآلي لاكتشاف الحالات الشاذة في النموذج الهجين.

بالنسبة للنموذج الأخير فهو يقسم مخرجات مرحلة استخدام خوارزمية البيز البسيطة المخفية في النموذج الهجين إلى عدة مجموعات صغيرة، باستخدام خوارزمية التجميع المعروفة (k-means)؛ وذلك من أجل تحسين أداء النموذج الهجين.

عدة قواعد بيانات متخصصة استخدمت لتعليم النظام وتقييم أداء الأنظمة المقترحة؛ وقد قمنا في هذه الدراسة بعمل مقارنة بين النماذج المقترحة و النماذج التقليدية وتبين تفوق الانظمة الهجينة المقترحة على النماذج والأنظمة التقليدية؛ وأن أداء النموذج الهجين المحسن عن طريق خوارزميات التجميع يفوق النماذج الهجينة والتقليدية.

Abstract

The growing number of network intrusive activities poses a serious threat to the reliability of network services. Businesses and individuals are suffering from these malicious interceptions. All network and web services are threatened by attacks, e-shopping, mail systems, bank services, governmental e-services, and so on. Intrusion Detection Systems (IDS) try to fill the vacuum in the security architecture since many intrusions are still undetected with other security techniques. Many studies try to find an optimal model for intrusion detection system with the best detection rate and lowest false alarm rate. Different machine learning techniques and algorithms employed in this field, Support Vector Machine, Neural Networks, Naïve Bayes, clustering algorithm, etc. Intrusion detection system is divided into anomaly detection and misuse detection. Misuse uses the known patterns to detect known attacks, and anomaly detection determines the outliers when they deviate from normal behavior. This study proposes three models for IDS, the first model employs one class classifier as anomaly detection system; the one class classifier algorithm based on the generation of artificial data from a reference distribution to form a two-class classification problem, and it combines the estimated reference density function with the class probability estimator to form an overall prediction; this algorithm can adapt any classification technique from the large number of classification algorithms for one class problems. The second model is hybrid model combines misuse and anomaly detection, where Hidden Naïve Bayes is used as for misuse detection and one class classifier for anomaly detection. One class support vector machine and one class classification algorithm applied in the first model are used for anomaly detection for our hybrid model. The last model decomposed the output of the misuse detection phase in hybrid model to smaller groups, using k-means clustering algorithm, in order to improve the performance of the hybrid model. KDDCup and NSL-KDD datasets are used to train and evaluate the proposed models. A comparison between the proposed models and conventional misuse and anomaly models shows that they are outperforms the conventional models; and the performance of the improved hybrid model is leading hybrid and conventional models.

Contents

1	Chapter 1 Introduction.....	1
	1.1 Security Fundamentals.....	2
	1.1.1 Assets.....	2
	1.1.2 Vulnerabilities.....	2
	1.1.3 Threats.....	2
	1.2 Challenges in Computer Security.....	4
	1.3 Intrusion Detection.....	5
	1.3.1 Motivations behind Intrusion Detection.....	6
	1.3.2 Goals of Intrusion Detection.....	6
	1.3.3 Types of Intrusion Detection.....	6
	1.4 Thesis Contribution.....	8
	1.5 Thesis Organization.....	9
2	Chapter 2 Background.....	10
	2.1 Literature View.....	10
	2.1.1 Misuse Detection.....	10
	2.1.2 Anomaly Detection.....	11
	2.1.3 Hybrid Models Detection:	11
	2.2 Discretization.....	12
	2.2.1 Entropy Minimization Discretization (EMD).....	13
	2.2.2 Proportional k-Interval discretization (PKID)	13
	2.3 Hidden Naïve Bayes algorithm.....	14
	2.4 One Class Classifier	17
	2.4.1 Evaluation Method.....	18
	2.4.2 The random forests algorithm.....	19
	2.5 One Class Support Vector Machine.....	20
3	Chapter 3 Proposed Frameworks	23
	3.1 Intrusion Anomaly Detection using One Class Classifier.....	23
	3.2 Intrusion Detection using Hybrid Model	23
	3.3 Improved Mixed Model using clustering.....	28
4	Chapter 4 Experimental Results.....	32
	4.1 Dataset.....	32
	4.2 Software and Tools	34
	4.3 Evaluation Measurement	34
	4.4 Data Preparation.....	35
	4.5 Result of Intrusion Anomaly Detection using One Class Classifier.....	36
	4.5.1 Parameter Optimization.....	36
	4.5.2 Evaluation and Discussion.....	38
	4.6 Results of Intrusion Detection using Hybrid Model	40
	4.6.1 Results of misuse detection.....	40
	4.6.2 Hybrid Framework with random forest one-class classifier	41
	4.6.3 Hybrid Framework with one-class SVM.....	50
	4.6.4 Comparison between Hybrid models.....	57
	4.7 Improved Mixed Model using clustering.....	60
	4.7.1 Parameter Optimization for Clustered Anomaly Detection.....	60
	4.7.2 Discussion.....	60
	4.7.3 Example of clusters.....	61
5	Chapter 5 Conclusion and Future Work.....	68
	References.....	70

List of Figures

Figure 1.1	Overview of Intrusion Detection System.....	5
Figure 2.1	Hidden Naive Bayes Structure.....	15
Figure 2.2	One-class Support Vector Machine.....	21
Figure 3.1	Block diagram of hybrid model.....	24
Figure 3.2	Proposed hybrid model.....	24
Figure 3.3	Training process of the proposed hybrid intrusion detection method.....	26
Figure 3.4	Testing process of the proposed hybrid intrusion detection method.....	27
Figure 3.5	Block diagram for improved hybrid model using clustering.....	28
Figure 3.6	Training process of the proposed improved hybrid intrusion detection method....	30
Figure 3.7	Testing process of the proposed improved hybrid intrusion detection method....	31
Figure 4.1	ROC Curve For 10%KDDCup with EMD obtained by OCC-RF.....	36
Figure 4.2	ROC Curve For 10%KDDCup with PKID obtained by OCC-RF.....	37
Figure 4.3	ROC Curve for NSL-KDDCup with EMD obtained by OCC-RF.....	37
Figure 4.4	ROC Curve for NSL-KDDCup with PKID obtained by OCC-RF.....	38
Figure 4.5	ROC curves of OCC-RF and one-class SVM anomaly detection models for NSL-KDD – EMD dataset.....	39
Figure 4.6	ROC curves of OCC-RF and one-class SVM anomaly detection models for NSL-KDD – PKID dataset.....	40
Figure 4.7	ROC Curve for HNB-OCC applied EMD-10% KDDCUP Dataset.....	43
Figure 4.8	ROC Curve for HNB-OCC applied to EMD-NSL-KDD Dataset.....	44
Figure 4.9	ROC Curve for HNB-OCC applied PKID-10% KDDCUP Dataset.....	45
Figure 4.10	ROC Curve for HNB-OCC applied to PKID-NSL-KDD Dataset.....	46
Figure 4.11	Comparison between results of EMD NSL-KDD model using different values of RBF Kernel g.....	50
Figure 4.12	ROC Curve for HNB-SVM applied EMD-10% KDDCUP Dataset.....	53
Figure 4.13	ROC Curve for HNB-SVM applied PKID-10% KDDCUP Dataset.....	54
Figure 4.14	ROC Curve for HNB-SVM applied to EMD-NSL-KDD Dataset.....	55
Figure 4.15	ROC Curve for HNB-SVM applied to PKID-NSL-KDD Dataset.....	56
Figure 4.16	Difference between HNB-OCC and HNB-SVM applied 10% KDDCUP- EMD Dataset.....	58
Figure 4.17	Difference between HNB-OCC and HNB-SVM applied 10% KDDCUP- PKID Dataset.....	58
Figure 4.18	Difference between HNB-OCC and HNB-SVM applied NSL-KDD EMD Dataset.....	59
Figure 4.19	Difference between HNB-OCC and HNB-SVM applied NSL-KDD PKID Dataset.....	59
Figure 4.20	Results of improved hybrid IDS for KDDCup EMD using HNB-OCC model....	63
Figure 4.21	Results of improved hybrid IDS for KDDCup EMD using HNB-SVM model....	64
Figure 4.22	Results of improved hybrid IDS for KDDCup PKID using HNB-OCC model....	64
Figure 4.23	Results of improved hybrid IDS for KDDCup PKID using HNB-SVM model....	65
Figure 4.24	Results of improved hybrid IDS for NSL-KDD EMD using HNB-OCC model...	65
Figure 4.25	Results of improved hybrid IDS for NSL-KDD EMD using HNB-SVM model...	66
Figure 4.26	Results of improved hybrid IDS for NSL-KDD PKID using HNB-OCC model...	66
Figure 4.27	Results of improved hybrid IDS for NSL-KDD PKID using HNB-SVM model ...	67

List of Tables

Table 2.1	Learning algorithm for HNB algorithm.....	32
Table 4.1	Mapping attack types to the attack classes on KDDCup 1999 dataset..	32
Table 4.2	Characteristics of KDDCup 1999 dataset (10% KDDCup version).....	33
Table 4.3	Confusion Matrix.....	34
Table 4.5	Results of misuse detection using HNB algorithm.....	40
Table 4.6	Optimal Value for hybrid model using OCC-RF.....	42
Table 4.7	DR and FAR for EMD 10% KDDCup when varied ntree and Mtry=1.	47
Table 4.8	DR and FAR for EMD NSL-KDD when varied ntree and Mtry=10....	48
Table 4.9	DR and FAR for PKID 10%KDDCup when varied ntree and Mtry=1	48
Table 4.10	Time estimation for hybrid models applied to NSL-KDD – EMD dataset.....	49
Table 4.11	Results of anomaly 1-class SVM and Hybrid IDS using HNB-SVM, for different value of RBF Kernal parameter g and $v = 0.01$	51
Table 4.12	Optimum parameters for one-class SVM hybrid	51
Table 4.13	Time comparison between different proposed hybrid models.....	52
Table 4.14	Parameter range used for improved hybrid IDS.....	61
Table 4.16	Example of improved hybrid system (HNB-OCC) using k-means clustering with $k = 10$, $Mtry = (0,1)$ and $ntree = (2,5,10,20,50,100)$ applied to NSL-KDD EMD.....	62
Table 4.16	Example of improved hybrid system (HNB-SVM) using k-means clustering with $k = 10$, $g = (0,0.1)$ and $v = (0.001,0.01,0.05,0.1,0.5)$ applied to KDDCup PKID.....	63

List of Abbreviations

IDS	Intrusion Detection System
OCC	One Class Classifier
RF	Random Forest Algorithm
SVM	Support Vector Machine
OCC-RF	One Class Classifier using Random Forest as class estimator
NB	Naïve Bayes
HNB	Hidden Naïve Bayes
DARPA	Defense Advanced Research Projects Agency
ACM	Association for Computing Machinery
KDD	Knowledge Discovery And Data Mining
DoS	Denial of Service
U2R	User to Root Attack
R2L	Remote to Local Attack
Prob	Probing Attack
RST	Rough Set Theory
RSVM	Robust Support Vector Machines
HID	Host Based IDS
NIDS	Network Based IDS
BPNN	Back Propagation Neural Network
DR	Detection Rates
FAR	False Positives
Reptree	Fast Decision Tree Learner
ADAM	Audit Data Analysis and Mining
NIDES	The Next-Generation Intrusion Detection Expert System
SOM	Self-Organization Map
OneR	One Rule Classifier
EMD	Entropy Minimization Discretization
PKID	Proportional K-Interval Discretization
MDL	Minimum Description Length
TAN	Tree Augmented Naive Bayes
SBC	Selective Bayesian Classifiers
ENB	Evolutional Naive Bayes
NBTree	Naive Bayes Tree
ROC	Receiver Operating Characteristic
Weka	Waikato Environment for Knowledge Analysis

Chapter 1

Introduction

With the evolution of technology, network-based services has become widely used to transfer sensitive information on a network between different types of computer devices, such as personal computers, laptops, Smartphones , iPads, tablets, etc. So security has become crucial issue for computer systems and networks. A secure network should provide data confidentiality, data integrity, and data availability. Intrusion is an action that tries to destroy data confidentiality, data integrity, and data availability of network information [1].

Although a wide range of security technologies such as access control, information encryption, and firewalls and intrusion prevention are used to protect network-based systems, there are still many undetected intrusions [2][3]. Given that, intrusion detection systems (IDSs) for automatic monitoring of network activities and detecting network attacks become an indispensable component of security infrastructure used to detect these threats before they inflict widespread damage [4][5].

When building an IDS one needs to consider many issues, such as data collection, data pre-processing, intrusion recognition, reporting, and response. Among them, intrusion recognition is at the heart. Audit data are examined and compared with detection models, which describe the patterns of intrusive or benign behavior, so that both successful and unsuccessful intrusion attempts can be identified [5].

There are two major intrusion detection methods: misuse detection and anomaly detection [5]. Misuse detection identifies intrusions by matching observed data with pre-defined descriptions of intrusive behavior. So well-known intrusions can be detected efficiently with a very low false positive rate. For this reason, the approach is widely adopted in the majority of commercial systems. However, intrusions are usually updatable, polymorph, and evolve continuously. So this type of detection will fail to detect novel or new intrusions. New update to model is needed to detect novel intrusions, update is done manually which is time consuming and laborious, or automatically with the using learning algorithms. Unfortunately, datasets for this purpose are usually expensive to prepare, as they require labeling of each instance in the dataset as normal or a type of intrusion. On the opposite side anomaly detection [6] is overcome the limitation of misuse detection. Anomaly detection algorithms analyze normal traffic and profile normal traffic patterns [7]; and detect outliers when they deviate from the normal patterns, anomaly detection is able to detect novel attacks but with higher false alarm rate [3][8]. To avoid disadvantages of misuse and anomaly detection techniques and maximize their advantages, there are a lot of proposed hybrid approaches proposed in the lase years such as [7][9].

Most hybrid detection systems independently train a misuse detection model and an anomaly detection model, and then simply aggregate the results of the detection models. For example, hybrid intrusion detection systems regard a traffic connection as an attack if at least one of the two models classifies the traffic connection as an attack. In this case, the detection rate will be improved but the IDS will still have a high false positive rate. In contrast, if the hybrid method regards a traffic connection

as an attack only if both models classify the connection as an attack, false alarms will be reduced but it may overlook many attack connections [7].

Many current IDSs are depends on a set of rules representing attacks or normal network characteristics, which are collected and identified by security experts [10]. Manually rule encoding is a higher cost process in time and money, and it depends on the efficiency of human experts in analyzing a huge amount of network activities to discover intrusion patterns. However, these drawbacks are overcome by employing many data mining techniques in IDSs [4][11-13].

Data mining is the analysis of observational datasets to discover the underlying models from a set of training data and to summarize the data in novel ways that are both understandable and useful to the data owner [14]. In fact, the process of automatically constructing models from data is not trivial, especially for intrusion detection problems. This is because intrusion detection faces such problems as huge network traffic volumes, highly imbalanced attack class distribution, the difficulty to realize decision boundaries between normal and abnormal behavior, and requiring continuous adaptation to a constantly changing environment [5]. Data mining techniques can be used in intrusion detection systems to classify network connections into intrusion and normal data based on labeled training data in misuse detection, and to group similar network connections together in clusters according to a given similarity measure in anomaly detection [15][16].

1.1 Security Fundamentals

There are three security fundamentals we should to study and understand, asset, vulnerability, and attack.

1.1.1 Assets

In an IT system, assets include Hardware, Software (applications, operating systems), Data and information (essential data for running and planning your business), and Reputation (the opinion held by your customers and the general public about your organization) [17][18].

1.1.2 Vulnerabilities

Vulnerabilities are weaknesses of a system that could be accidentally or intentionally exploited to damage assets [18]. An example is accounts with root privileges where the default password, such as 'root', has not been changed. Granting full control access to everyone for a shared folder is another example.

1.1.3 Threats

A threat is a potential violation of security. The violation need not actually occur for there to be a threat. The fact that the violation might occur means that those actions that could cause it to occur must be guarded against (or prepared for). Those actions are called attacks[19]. So Network attacks are defined as a set of malicious activities try to exploit vulnerabilities to disrupt, deny, degrade or destroy information and service resident in computer networks (assets) . A network attack is executed through the data stream on networks and aims to compromise the Integrity, Confidentiality or Availability of computer network systems. Network attacks can vary from annoying email directed at an individual to intrusion attacks on sensitive data, computer

information systems and critical network infrastructure [18] [20]. The one who execute such activities, or cause them to be executed, are called attackers

Threats can be categorized in many ways [21-23]. Microsoft's STRIDE threat model [22] is an example that categorize threats by the damage done to assets, STRIDE lists the following categories.

1. Tampering with data: Security settings are changed to give the attacker more privileges.
2. Spoofing identities: The attacker pretends to be somebody else.
3. Repudiation: A user denies having performed an action like making a purchase.
4. Information disclosure: Information might lose its value if it is disclosed to the wrong parties (e.g., trade secrets).
5. Denial of service (DoS): DoS attacks can make websites temporarily unavailable.
6. Elevation of privilege: The term elevation of privilege refers to a user who gains more privileges on a computer system than he or she is entitled to.

Another categorization is used in KDDCup dataset where attacks fall in one of the following four categories [21]:

1. Denial of Service Attack (DoS): is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine. e.g. syn flood
2. User to Root Attack (U2R): is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system. e.g., various "buffer overflow" attacks
3. Remote to Local Attack (R2L): occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine. e.g. guessing password
4. Probing Attack: is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls e.g., port scanning

Computer security involves the protection of assets. The three security services confidentiality, integrity, and availability counter threats to the security of a system [19]. Security can be expressed as a function of the three security services. Ideally, security professionals attempt to maximize the security depending on the security requirement of an organization or application [17].

Confidentiality

Confidentiality deals with the secrecy or privacy of assets. It ensures that only authorized users are allowed to access computer assets. This 'access' incorporates any kind of access including reading, writing, printing or even the knowledge that a particular asset exists [17]. In short, confidentiality means that only authorized people or systems can access protected data.

✚ Integrity

Integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized modification, modification of an asset may include tasks like changing, deleting, writing, changing status, and creating. Integrity includes data integrity (the content of the information) and origin integrity (the source of the data, often called authentication) [17][19]. According to Clark and Wilson [23], integrity is maintained when "No user of the system, even if authorized, may be permitted to modify data items in such a way that assets or accounting records of the company are corrupted".

According to the Orange Book [24], integrity may also be defined as ensuring external consistency.

An example of integrity violation is when data is transmitted across a network. An attacker could intercept and modify packets of data on the network if that data's integrity is not protected. This type of attack is known as a man-in-the-middle attack [18].

✚ Availability

Availability refers to the ability to use the information or desired resource to authorized users whenever desired [17][19]. Availability is an important aspect of reliability as well as of system design because an unavailable system is at least as bad as no system at all [19]. An Attempts to block availability, called denial of service attacks.

Apart from the above three main properties, there are other properties which may be considered a part of computer security. These include authentication, accountability, reliability, fault-tolerance and assurance

1.2 Challenges in Computer Security

Ideally, a computer system can be made perfectly secure if all the above mentioned properties are well satisfied. However, in practice it is impossible to design a system with perfect security and usability [25]. Any system can be subjected to breaches of confidentiality, integrity and/or availability thereby rendering itself in an insecure state. In order to address this scenario, it is acknowledged that a system might fail and so there is a need to put in detection and response mechanisms in addition to the protection mechanisms [25].

✚ Protection

The proactive part of security consists of protecting the asset. The asset is protected in order to prevent any violation of confidentiality, integrity or availability [17].

✚ Detection

Since perfect security cannot be achieved, we anticipate that the protection measures might not be able to protect the assets under all cases. This leads to the adoption of detection measures in security. These measures are used detect possible violation of security and their efficacy depends on the time taken to

detect. This time may be different for different assets and may be proportional to the value of the asset. Another factor that contributes to the efficiency of a detection mechanism is the number of false alarms it generates. A false alarm may be a false positive or a false negative. The higher the number of false alarms, the slower is the detection process and is more expensive.

✚ Response

Supplementing the detection process is the process of responding to security violation. The response type may be different in different scenarios and would depend on the exact security requirement. Typical response types include evaluating the damage, recovering from the damage, etc.

1.3 Intrusion Detection

Intrusion is a set of actions that attempt to violate the integrity, availability or confidentiality of data on a computing platform [17].

An intrusion detection system (IDS) is a software, hardware, or hybrid of both that used to detect violation of a security policy of an organization. These violations may be caused by people external to the organization (i.e. attackers) or by employees of the organization (i.e. insiders). IDS functions as “Radar” to monitor all the network and system activities [1][17], and decides whether these activities are symptomatic of an attack or constitute a legitimate use of the system [26]. The main objective of IDS is to alarm the system administrator if any suspicious activity happening. Figure 1.1 illustrates the organization of an IDS [5] where solid arrows indicate data and control flow while dotted arrows indicate a response to intrusive activities.

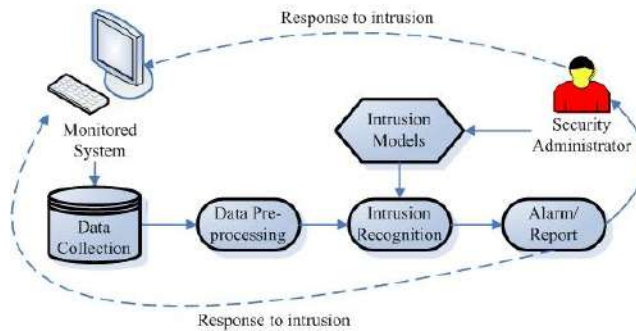


Figure 1.1 Overview of IDS

An intrusion detection system completes its task through classification and determination rules that are composed of two main parts: a pre-existing knowledgebase and a set of classification algorithms. The pre-existing knowledge base mainly includes (1) a set of estimated parameters based on the previous network traffic, (2) a set of known and labeled attacks or anomaly events, and (3) a group of data sources. The classification algorithms are mainly constructed based on either statistical modeling, Artificial Intelligence, or a hybrid of both. The performance of intrusion detection systems is affected by different factors such as the quality of the pre-existing knowledgebase, the robustness of the classification algorithms, and the uniqueness of

attacks or anomalous events. The pre-existing knowledgebase should be updated regularly so that any newly discovered attacks or malicious codes can be labeled and the classification parameters revised accordingly.

IDS aims to detect as many type of attacks as possible including those by attackers and those by insiders, with minimum number of false alarms in the best possible time [17].

1.3.1 Motivations behind Intrusion Detection

Intrusion Detection has received considerable motivation owing to the following reasons: [17][27]

1. If an intrusion is detected quickly enough, an intruder can be identified quickly and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less is the amount of damage done and more quickly that recovery can be achieved.
2. An effective intrusion detection system can serve as a deterrent, an alert is send by IDS to security officer to take the appropriate action, also IDS consider the core to intrusion prevent systems.
3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

1.3.2 Goals of Intrusion Detection

Along with the motivations, the goals of intrusion detection can be summarized as below [17][19]:

- ✚ Detect as many type of intrusions as possible. Intrusions from within the site, as well as those from outside the site, are of interest. Furthermore, both known and previously unknown attacks should be detected.
- ✚ Detect intrusions in a timely fashion. "Timely" here need not be in real time. Often, it suffices to discover an intrusion within a short period of time. Real-time intrusion detection raises issues of responsiveness. If every command and action must be analyzed before it can be executed, only a very simple analysis can be done before the computer (or network) being monitored becomes unusable.
- ✚ Present the analysis in a simple, easy-to-understand format.
- ✚ Detect as accurately as possible thereby minimizing the number of false alarms.

1.3.3 Types of Intrusion Detection

Based on their functionality (classification rule), these techniques can be classified into misuse and anomaly detection.

A misuse detection system or signature-based IDS looks at the packets in network traffic and compares network activities with predefined signatures or patterns taken from characteristic features that represent a specific attack, and issues an alert if any suspicious activity has been identified [3][28]. So that detecting new attacks require

new rules and such a system can discover attacks with a low false positive rate, but it cannot discover novel attacks[8][17][29]. Misuse model are standardized and understandable by security personnel [18].

Modeling of misuse requires a knowledge of system vulnerabilities or potential vulnerabilities that attackers attempt to exploit. The intrusion detection system incorporates this knowledge into a rule set. When data is passed to the intrusion detection system, it applies the rule set to the data to determine if any sequences of data match any of the rules. If so, it reports that a possible intrusion is underway. Misuse-based intrusion detection systems often use expert systems to analyze the data and apply the rule set [19].

A weakness of the misuse-based IDS approach is the failure to characterize slow attacks that extend over a long period of time. To identify these types of attacks, large amounts of information must be held for extended time periods. Another issue that the IDS is resource-intensive. The knowledge database continually needs maintenance and updating with new vulnerabilities and environments to remain accurate [18].

Anomaly detection discovers attacks by identifying deviations from normal network activities or anomaly-free behavior patterns [3][28]. In this type, observable behaviors of a system are used to build models for normal system operation. These behaviors may include audit logs, network sensors, system calls, etc. [17]. There are two assumptions that the abnormal behavior is rare and it is different from the normal behavior [5][30]. However, these assumptions are not always true because of the high degree of similarity between some kinds of normal and intrusions connections, which makes these intrusions to stick to normal data in the same class causing very high false positive rates [3].

The main advantage of this type is that anomaly detection can discover novel attacks but with a high false positive rate, but it is difficult in practice because it is hard to define an anomaly-free (normal behavior) pattern [12][28]. Another difficulty is that activity and behavior of the users of a networked system might not be consistent enough to effectively implement a behavior-based IDS [18]. However, its major difficulty lies in discovering boundaries between normal and abnormal behavior, due to the deficiency of abnormal samples in the training phase [5].

Earlier studies [6][19] identifies three different anomaly models; threshold-based, profile-based and Markov model. The first model uses a threshold metric. The threshold-based measures the frequency of anomalous events in a specified period [28]. A minimum of m and a maximum of n events are expected to occur (for some event and some values m and n). If, over a specific period of time, less than m or more than n events occur, the behavior is deemed anomalous.

The second model is profile-based, profile-based model focuses on analyzing current or historical user behavior and detects any outlier values based on a series of measures such as mean, median, standard deviation, or interval estimates of various user activity-related parameters and variables (e.g., login time zone, length of connections, types of protocol, connection statuses) [28]. The profile based model provides more flexibility than the threshold model. Administrators can tune it to discriminate better than the threshold model. But with flexibility comes complexity. In particular, an explicit assumption is that the behavior of processes, and users, can be statistically modeled. If this behavior matches a statistical distribution (such as a gaussian or

normal distribution), determining the parameters requires experimental data that can be obtained from the system. But if not, the analysts must use other techniques, such as clustering, to determine the characteristics, and the values that indicate abnormal behavior [19].

The third model is a Markov model. Examine a system at some particular point in time. Events preceding that time have put the system into a particular state. When the next event occurs, the system transitions into a new state. Over time, a set of probabilities of transition can be developed. When an event occurs that causes a transition that has a low probability, the event is deemed anomalous. This model suggests that a notion of "state," or past history, can be used to detect anomalies. The anomalies are now no longer based on statistics of the occurrence of individual events, but on sequences of events.

We can classify Intrusion Detection Systems according to protected system to network-based, host-based, and Hybrid system. IDSs that operate on a specific host and detect malicious activity on that host only are called host-based IDS (HID). IDSs that operate on network segments and analyze that segment's traffic are called network based IDS (NIDS). Because there are pros and cons of each, an effective intrusion detection strategy should use a combination of both network-based and host-based IDSs.

1.4 Thesis Contribution

Various techniques and frameworks were proposed in the last decade in the field of intrusion detection, were trying to find the optimal solution (accuracy, precision, detection rate, time consumption). In our research we propose a new anomaly detection model and hybrid models that try to improve the performance of intrusion detection system. Three contributions are proposed in this study; first an anomaly detection model using one class classifier algorithm (OCC) [31], with Random Forest (RF) as probability estimation and gaussian density as density estimator. OCC-RF is used to profile normal instances, it generates artificial data to represent other data class. OCC-RF anomaly detection model is compared to conventional one class support vector machine (one-class SVM), the comparison shows that OCC-RF model outperforms conv. one-class SVM model.

The second model is hybrid IDS using new combination of misuse and anomaly detection. Hybrid IDS employs Hidden Naïve Bayes (HNB) algorithm in misuse detection phase, HNB is used to classify network connections into intrusion and normal data based on a labeled training dataset that helps in building classification patterns; in anomaly detection phase an outlier detection algorithms are used, OCC-RF and one-class SVM are used, each one is used individually with HNB. Hybrid model is used to improve the performance (especially detection rate and false alarm rate) of the intrusion detection system; and to overcome the limitations of misuse and anomaly detection systems. A comparison between the proposed hybrid models and conventional anomaly intrusion detection models using the same algorithms are performed, also a comparison between different proposed models is done to show the best model. The results show good performance of the proposed hybrid models.

The last model uses k-means clustering algorithm as an improvement to the second model. After using HNB algorithm, instances classified as normal are grouped into k clusters based on the similarity of connections features. Since each cluster will contains partition of total instances and these instances are homogeneous and similar, building models for each cluster of outlier detection using one-class SVM or OCC are simpler, easier and faster than dealing with whole data one time as in hybrid IDS. Results show that the improved model using clustering is leading the proposed hybrid model.

In our study the proposed models are evaluated over a real network connections data which are generated from the Defense Advanced Research Projects Agency (DARPA) network connections, which was prepared by ACM Special Interest Group on Knowledge Discovery and Data Mining in the Knowledge Discovery and Data Mining 1999 (KDDCup'99) contest. KDDCup dataset suffers from some problems [21], KDDCup is imbalanced and contains redundant connections and this cause the learning algorithms and the evaluation results to be biased towards the frequent records. NSL-KDD is a modified version of KDDCup that tries to alleviate limitations without distorts the data shape. We evaluate our proposed models using 10%KDDCup (a reduced version) and NSL-KDD (improved version) datasets.

1.5 Thesis Organization

The study is organized as follows. Chapter 2 discuss related works and theoretical background for the used artificial intelligence algorithms. Chapter 3 presents the proposed models; anomaly detection model, hybrid model, and improved hybrid model. Chapter 4 shows and discussed the experimental results. Finally, Chapter 5 summarizes the study and discusses future work.

Chapter 2

Background

In this chapter, we take fast review to previous works in the field of IDSs, and we discuss a background about artificial intelligence algorithms used in this study.

2.1 Literature Review

Most of artificial intelligence and data mining techniques has been introduced to develop intrusion detection systems, old researches in this field concentrating in using the algorithm itself or one of its improved versions and apply it to IDS, recent researches uses combinations of these algorithms to get improvement in performance (detection rate, accuracy, and false alarm). As we motioned in the previous chapter that IDSs are divided into misuse detection and anomaly detection, here we present samples of previous work for each type:

2.1.1 Misuse Detection

Hu et al [32] compared the performance of robust support vector machines intrusion detection system (RSVMs) with that of conventional support vector machines and nearest neighbor classifiers in separating normal usage profiles from intrusive profiles of computer programs. The results indicate the superiority of RSVMs not only in terms of high intrusion detection accuracy and low false positives but also in terms of their generalization ability in the presence of noise and running time. Chen et al [33] proposed RST (Rough Set Theory) and SVM model to detect intrusions. First, RST is used to preprocess the data and reduce the dimensions. Next, the features selected by RST will be sent to SVM model to learn and test respectively. The method is effective to decrease the space density of data and to improve the false positive rate and accuracy.

Complicated model was proposed by Chandrasekhar et al [34], which combines multiple artificial intelligent algorithms such as neuro-fuzzy and radial basis support vector machine (SVM) for helping IDS to attain higher detection rate. The proposed technique has four major steps: primarily, k-means clustering is used to generate different training subsets. Then, based on the obtained training subsets, different neuro-fuzzy models are trained. Subsequently, a vector for SVM classification is formed and in the end, classification using radial SVM is performed to detect intrusion has happened or not. Experimental results shows that the proposed approach do better than BPNN, multiclass SVM and other well-known methods such as decision trees and Columbia model in terms of sensitivity, specificity and in particular detection accuracy.

Naïve Bayes classifier and its enhanced model was presented to IDS also. A new learning algorithm for adaptive network intrusion detection using naive Bayesian classifier and decision tree is presented by Dewan et al [35], The experimental results prove that the proposed algorithm achieved high detection rates (DR) and significant reduce false positives (FP) for different types of network intrusions using limited computational resources. NBTree classifier that combines decision tree with Naïve Bayes classifier was applied by Sabnani [17] to intrusion detection system. Over 99% accuracy is obtained by it in case of feature selection and without using it. Koc et al

[36] applied Hidden Naïve Bayes (HNB) model to intrusion detection problems and compared it with the traditional Naïve Bayes model. The results show that the HNB model exhibits a superior overall performance in terms of accuracy, error rate and misclassification cost compared with the traditional Naïve Bayes model, leading extended Naïve Bayes models and the Knowledge Discovery and Data Mining (KDD) Cup 1999 winner. Also it's performed better than other leading state-of-the art models, such as SVM, in terms of accuracy.

2.1.2 Anomaly Detection

Eskin et al. [12] investigate three algorithms for unsupervised anomaly detection: cluster-based estimation, k-nearest neighbor, and one-class support vector machine (SVM). Other studies uses clustering approaches in unsupervised IDSs [11][37]. Supervised anomaly detection uses attack-free training data to build profiles of normal activities. After that, it uses the deviation from the profiles to detect intrusions. Supervised anomaly detection has been studied extensively such as fuzzy data mining and genetic algorithms [38], neural networks [29], and SVM [39][40].

Agarwal et al [40] proposed anomaly traffic detection system based on the Entropy of network features and Support Vector Machine (SVM). Agarwal et al [40] proved that entropy based detection technique is capable of identifying anomalies in network better than support vector machine based detection system. In addition, hybrid approach used both entropy of network features and support vector machine outperforms entropy and SVM based techniques.

Sindhu et al. [41] proposed a Decision Tree based light weight intrusion detection using a wrapper approach for anomalies detection in network. The proposed method has evaluated using detection percentage and error percentage. The proposed method gave better results as compare to Decision Stump, C4.5, Naive Bayes, Random Forest, Random Tree, and REPTree.

2.1.3 Hybrid Models Detection

Many researches are performed on hybrid intrusion detection methods in the last three years that attempts to overcome the limitations of the anomaly detection and misuse detection methods.

Audit Data Analysis and Mining (ADAM) [42] is one of the most widely known projects in the area of data-mining-based intrusion detection. ADAM uses a combination of association rule mining and a classification method to detect attacks. The framework of ADAM has two phases: a training phase and an online phase. In the training phase, the attack-free training data are fed to a module that performs offline association rule discovery. The output of this module is a rule-based profile of normal activities. After that, the labeled training data are fed into a classifier builder to train the classifier. In the online phase, the test data are fed into the system. With the built profile, the system finds the items classified as false alarms, attacks, and unknown attacks by the trained classifier. The unknown attacks are the suspicious items that cannot be classified as false alarms or attacks.

The next-generation intrusion detection expert system (NIDES) [43] is another example of hybrid IDS. NIDES combining a misuse and anomaly detection to increase the chances to detect intrusions. NIDES performs real-time monitoring of user activity on multiple-target systems connected on a network. The rule-based misuse detection component employs expert rules to define known intrusive activities, and the anomaly detection component is based on a statistical approach.

Depren et al [9] proposed an intelligent hybrid intrusion detection system that consists of an anomaly detection model, a misuse detection model, and a decision support system. They modeled the anomaly detection model with a self-organization map (SOM) and the misuse detection model with a decision tree. Each model is trained independently, and then the decision support system simply combines the classification results of both models. Proposed method is evaluated using parameter data rate and false positive rate. Proposed method gave detection rate of 98.96% and a false positive rate of 1.01% for anomaly detection module and also a classification rate of 99.61% and a very low false positive rate of 0.20% are achieved for the misuse detection module.

Kim et al [7] combined decision tree with one-class SVM in a novel model, a misuse detection model is built based on the C4.5 decision tree algorithm and then the normal training data is decomposed into smaller subsets using the model. Next, multiple one-class SVM models are created for the decomposed subsets. The experimental results demonstrate that the proposed method is better than the conventional methods in terms of the detection rate for both unknown and known attacks while it maintains a low false positive rate. In addition, it reduces time complexity, the training and testing time of the anomaly detection model is shown to be only 50% and 60%, of the time required for the conventional models. Reda et al [3] proposed a hybrid detection framework that depends on data mining classification and clustering techniques, the random forest algorithm is used into a misuse detection method. And the k-means algorithm is used as unsupervised anomaly detection method to partition the captured network connections into a specified number of clusters, and then detect the anomalous clusters depending on their features. The proposed hybrid method gives good result comparing to traditional techniques. Muda et al [44] proposed approach that combines the k-means clustering with the OneR classification technique, by combining clustering (to identify groups of similarly behaved samples, i.e. malicious and non-malicious activity) and classification techniques (to classify all data into correct class categories), the result shows that proposed approach achieve a better accuracy and detection rate, particularly in reducing the false alarm.

2.2 Discretization

Discretization as a pre-processing step is widely used to improve the performance of the IDS as shown in many proposed IDS, as shown in [36][45][46]. Discretization is used to partition or convert continuous attributes features or variables to nominal attributes, features or variables with a finite number of values [45]. Better models can be produced by discretization of continuous attributes, and this will improve the accuracy of classifiers, including Naïve Bayes classifiers, especially in larger datasets as shown in earlier studies [47][48]. Front-end discretization might be necessary for

some classifiers if their algorithms cannot handle continuous features by design like HNB [49].

Numerous studies such as [48],[49],[51], and [52] have examined discretization methods in the last two decades to determine how continuous values should be grouped, how cut points should be positioned on the continuous scale and how many intervals should be used to generate datasets.

We improve the performance of our proposed frameworks using two leading discretization methods during the pre-processing phase: entropy minimization discretization (EMD) and proportional k-interval discretization (PKID). These two methods are selected because of their performance on large datasets, particularly the KDDCup'99 dataset [36][53][54].

2.2.1 Entropy Minimization Discretization (EMD)

EMD [50] evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For evaluating each candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidates. The binary discretization is applied recursively, always selecting the best cut point until the stopping condition, which is based on the minimum description length (MDL) method is reached [47][50].

2.2.2 Proportional k-Interval discretization (PKID)

Tunes the interval size and interval number proportional to the number of training instances to find an appropriate trade-off between the granularity of the intervals and the expected accuracy of the probability estimation. The trade-off can also be observed as a trade-off between discretization bias and variance [36]. The larger the interval size (the smaller the interval number), the lower the variance but the higher the bias. In the converse, the smaller the interval size (the larger the interval number), the lower the bias but the higher the variance. Lower learning error can be achieved by tuning the interval size and number to find a good trade-off between the bias and variance [50].

Equal weights are given initially to bias and variance by creating square root of n intervals with square root of n instances in each interval, where n is the number of instances for a continuous feature [36][50]. As n increases, both discretization bias and variance can decrease. This means that PKID has greater capacity to take advantage of the additional information inherent in large volumes of training data. Previous experiments [54] showed that PKID significantly reduced classification error for larger datasets. But PKID was sub-optimal for smaller datasets. This might be because when n is small, PKID tends to produce a number of intervals with small size which might not offer enough data for reliable probability estimation, thus resulting in high variance and poor performance of Naïve Bayes classifiers[50].

2.3 Hidden Naïve Bayes algorithm [49]

In order to relax the conditional independence assumption of naïve Bayes effectively, an appropriate language and efficient machinery to represent and manipulate independence assertions are needed. In order to avoid the intractable complexity for learning Bayesian networks, learning-improved Naïve Bayes has attracted much attention from researchers. Related works can be divided into five main categories:

1. Structure extension. Extending the structure of naïve Bayes by using directed arcs to represent the dependencies among attributes, TAN (Tree-augmented naïve Bayes) is an example of structure extension work.
2. Feature selection. Removing redundant and/or irrelevant attributes from training data sets and only selecting an attribute that are the most informative in learning such as : selective Bayesian classifiers (simply SBC) and evolutionary naïve Bayes (ENB)
3. Attribute weighting. Assigning different weights to attributes in building naïve Bayes. Paper [55] uses various attribute weighting methods: the gain ratio method, the hill climbing method, the Markov Chain Monte Carlo method, and combination of these methods.
4. Local learning. Employing the principle of local learning to build a local naïve Bayes. Naïve Bayes tree (NBTree) [56] is an example.
5. Data expansion. Expanding training data and building a naïve Bayes on the expanded training data.

Hidden Naïve Bayes is a structure-extension-based algorithm used to relax the conditional independence assumption of naïve Bayes. In HNB, the structure of naïve Bayes is extended by creating a layer of hidden parents. Thus, the resulting structure is more complex than naïve Bayes. In that sense, it is similar to TAN.

The idea of HNB is to create a hidden parent for each attribute, which combines the influences from all other attributes. In this way HNB avoid the computational complexity for learning an optimal Bayesian network and still take the influences from all attributes into account.

Figure 2.1 gives the structure of an HNB. In Figure 2.1, C is the class node, and is also the parent of all attribute nodes. Each attribute A_i has a hidden parent A_{hpi} , $i = 1, 2, \dots, n$, represented by a dashed circle. The arc from the hidden parent A_{hpi} to A_i is also represented by a dashed directed line, to distinguish it from the regular arcs.

The joint distribution represented by an HNB is defined as follows:

$$P(A_1, \dots, A_n, C) = P(C) \prod_{i=1}^n P(A_i | A_{hpi}, C) \quad (1)$$

Where

$$P(A_i | A_{hpi}, C) = \sum_{j=1, j \neq i}^n W_{ij} * P(A_i | A_j, C) \quad (2)$$

The hidden parent A_{hpi} for A_i is essentially a mixture of the weighted influences from all other attributes. The classifier corresponding to an HNB on an example E is defined as follows:

$$E = (a_1, \dots, a_n)$$

$$c(E) = \arg \max_{c \in C} P(c) \prod_{i=1}^n P(a_i | a_{hpi}, c) \quad (3)$$

Where

$$P(a_i | a_{hpi}, c) = \sum_{j=1, i \neq j}^n W_{ij} * P(a_i | a_j, c) \quad (4)$$

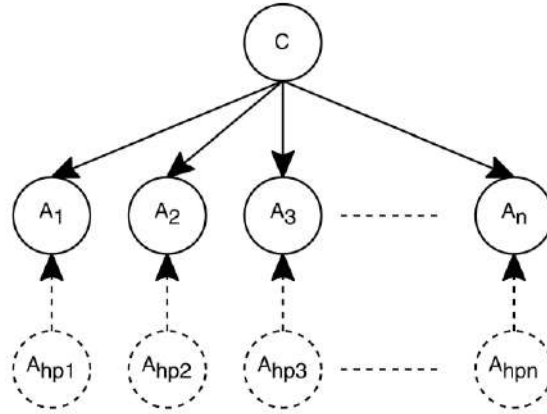


Figure 2.1 Hidden Naive Bayes Structure

In an HNB, attribute dependencies are represented by hidden parents of attributes. The way of defining hidden parents determines the capability of representing attribute dependencies. One-dependence estimators $P(A_i | A_j, C)$ are used in (2) to define hidden parents.

The importance of an attribute is determined by a weight W_{ij} . The approach to determining the weights W_{ij} is crucial for learning an HNB as we see from (1) and (2). There are two general approaches to doing it: performing a cross-validation-based search, or directly computing the estimated values from data. We use the second approach, and use the conditional mutual information between two attributes A_i and A_j as the weight of $P(A_i | A_j, C)$. So W_{ij} is defined as:

$$W_{ij} = \frac{I_P(A_i; A_j | C)}{\sum_{j=1, j \neq i}^n I_P(A_i; A_j | C)} \quad (5)$$

Where $I_P(A_i; A_j|C)$ is the conditional mutual information defined as:

$$I_P(A_i; A_j|C) = \sum_{a_i, a_j, c} P(a_i, a_j, c) \log \frac{P(a_i, a_j|c)}{P(a_i|c)P(a_j|c)} \quad (6)$$

Hidden Naïve Bayes estimating the parameters directly from the training data, the learning algorithm is shown in Table 2.1

Table 2.1 Learning algorithm for HNB algorithm [49]

Hidden Naïve Bayes (D)
Input: a set D of training examples Output: a hidden naïve bayes for D
for each value c of C compute $P(c)$ from D
for each pair of attributes A_i and A_j for each assignment $a_i, a_j, \text{ and } c$ to $A_i, A_j \text{ and } C$ compute $P(a_i; a_j c)$ from D
for each attribute A_i and A_j compute $I_P(A_i; A_j C)$ and W_{ij} from D

A three-dimensional table of probability estimates for each attribute value, conditioned by each other attribute value and each class, is generated, as shown in Table 2.1. And the conditional mutual information $I_P(A_i; A_j|C)$ for each pair of attributes is needed to compute to create the hidden parent of an attribute.

Probabilities $P(c)$ and $P(a_i; a_j|c)$ are estimated in [49] using the M-estimation as in the following equations:

$$P(c) = \frac{F(c) + 1.0/k}{t + 1.0} \quad (7)$$

$$P(a_i|a_j, c) = \frac{F(a_i, a_j, c) + 1.0/n_i}{F(a_j, c) + 1.0} \quad (8)$$

Where F is the frequency with which a combination of terms appears in the training data, t is the number of training examples, k is the number of classes, and n_i is the number of values of attribute A_i .

HNB was tested in terms of classification accuracy, using different data sets, and compare it to naïve Bayes (NB), selective Bayesian classifiers (SBC), naïve Bayes tree (NBTree), tree-augmented naïve Bayes (TAN), and averaged one-dependence estimators (AODE). The experimental results of [37] and [50] show that HNB significantly outperforms NB, SBC, NBTree, TAN, and AODE.

2.4 One Class Classifier [31]

The one class classifier algorithm presented by [31] in 2008. Which based on the generation of artificial data from a reference distribution to form a two-class classification problem. However, unlike earlier work on using artificial data for one-class classification, it is based on using a two-class probability estimator, and combines the estimated reference density function with the resulting class probability estimator to form an overall prediction.

The algorithm is generic in the sense that it can applied using an arbitrary density estimator and an arbitrary class probability estimation technique [31]; so from large number of classification algorithms we able to adapt them for one class problems.

Let T denote the target class for which we want to build a one-class model. We have training data for this class. Let A be the artificial class, for which we generate artificial data using a known reference distribution. Let X denote an instance and let $P(X|A)$ denote the density function of the reference distribution.

What we would like to obtain is $P(X|T)$, the density function for the target class. If we had this density function, we could use it for one-class classification by imposing a threshold on its values. In practice, we need to estimate this function using a class probability estimator learned from the training data.

The following shows how we can compute the density function for T , namely $P(X|T)$, given the class probability function $P(T|X)$, the reference density $P(X|A)$, and $P(T)$, which is the prior probability of observing an instance of the target class.

We start with Bayes' theorem:

$$P(T|X) = \frac{P(X|T) P(T)}{P(X)} \quad (9)$$

For a two-class situation, the probability of X is the probability of seeing an instance of X with either class label, so the equation becomes:

$$P(T|X) = \frac{P(X|T) P(T)}{P(X|T) P(T) + P(X|A) P(A)} \quad (10)$$

Now we solve for $P(X|T)$, the density function for the target class, which we want to use for one-class classification. We first bring the denominator on the right to the left:

$$(P(X|T)P(T) + P(X|A)P(A))P(T|X) = P(X|T)P(T) \quad (11)$$

Now we expand the product on the left, and bring the term involving $P(X|T)$ to the right:

$$P(X|T)P(T)P(T|X) + P(X|A)P(A)P(T|X) = P(X|T)P(T) \quad (12)$$

$$P(X|A)P(A)P(T|X) = P(X|T)P(T) - P(X|T)P(T)P(T|X) \quad (13)$$

Then we extract out $P(X|T)$ and bring the remainder to the left:

$$P(X|A)P(A)P(T|X) = P(X|T)(P(T) - P(T)P(T|X)) \quad (14)$$

$$\frac{P(X|A)P(A)P(T|X)}{P(T) - P(T)P(T|X)} = P(X|T) \quad (15)$$

We swap the two sides and extract $P(T)$ in the denominator:

$$P(X|T) = \frac{P(X|A)P(A)P(T|X)}{P(T)(1 - P(T|X))} \quad (16)$$

Now we make use of the fact that $P(A) = 1 - P(T)$, because there are only two classes, and rearrange:

$$P(X|T) = \frac{(1 - P(T))P(T|X)}{P(T)(1 - P(T|X))} P(X|A) \quad (17)$$

This equation relates the density of the artificial class $P(X|A)$ to the density of the target class $P(X|T)$ via the class probability function $P(T|X)$ and the prior probability of the target class $P(T)$.

To use this equation in practice, we choose $P(X|A)$ and generate a user-specified amount of artificial data from it

Target dataset (T) and artificial datasets (A) are then combined. The proportion of instances belonging to T in this combined dataset is an estimate of $P(T)$, and we can apply a learning algorithm to this two-class dataset to obtain a class probability estimator that takes the role of $P(T|X)$.

We can empirically choose an appropriate threshold on $P(X|T)$ to perform classification, and we can adjust this threshold to tune the probability of an instance being identified as an outlier.

we can apply any density estimation technique to the target data for class T and use the resulting density function to model the artificial class A . The more accurately this initial density estimate models $P(X|T)$, i.e. the better the match between $P(X|A)$ and $P(X|T)$, the easier the resulting two-class class probability estimation task should become.

In practice, given the availability of powerful methods for class probability estimation, and the relative lack of such techniques for density estimation, it makes sense to apply a simple density estimation technique to the target data first, to obtain $P(X|A)$, and then employ a state-of-the-art class probability estimation method to the two-class problem that is obtained by joining the artificial data generated using $P(X|A)$ and the data from the target class [31].

2.4.1 Evaluation Method

The one-class classification method presented combines the output of a density estimator with that of a class probability estimator. In our evaluation we use random forest as the probability estimator $P(T|X)$. Ten iterations were used throughout. Random forest give flexibility and can estimate probability with different models that can be built by utilizing its parameters. We evaluated a simple density estimation models, a gaussian density with a diagonal co-variance matrix containing the observed variance of each attribute in the target class. The amount of artificial data generated

using the reference distribution, which determines the estimate of $P(T)$ in equation (17), was set to 50% of the size of the target class.

In this study we evaluating one-class classifiers on KDDCup dataset, dataset in multiple classes. Instead of dealing with many classes we reconstruct the data for two classes as recommended by [57]. "Normal" class that is treated as target class and other classes are joined into an "anomaly" class.

2.4.2 The Random Forests algorithm

The random forests algorithm (RF) is a classification algorithm consisting of a collection of tree structured classifiers, where each tree casts a unit vote for the most popular class at each input [3][58].

A random forest is an ensemble (i.e., a collection) of unpruned decision trees. RF are often used when we have very large training datasets and a very large number of input variables. A random forest model is a classifier that consists of many decision trees and outputs the class that is the mode of the class output by individual trees [58][59].

Each tree is constructed using the following algorithm: [3][59]

1. Let the number of training cases be N , and the number of variables in the classifier be M .
2. We are select number m of input variables to be used to determine the decision at a node of the tree; m should be much less than M .
3. Choose a training set for this tree by choosing N times with replacement from all N available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
4. For each node in the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set. The value of m is held constant during the forest growing.
5. Each tree is grown to the largest extent possible [58]. There is no pruning of the trees in the RF algorithm. Instead, the RF creates a set of largely uncorrelated trees, and combines their results to form a generalized predictor.

There are two important parameters that significantly affect the error rate of the RF algorithm: the number of random features used to split each tree node ($Mtry$), and the number of trees grown in the forest (n_{tree}). To achieve a good performance and a low error rate, these parameters are optimized by building the forest with different $Mtry$ and n_{tree} values to choose the best values that achieve the lowest error rate with highest detection rate.

The most important parameter to choose is $Mtry$, the number of input variables tried at each split, it has been reported that the default value is often a good choice [60][61]; Where the default value is equals to $int((\log_2 \#inputs) + 1)$ [58].

The error rate of a forest depends on the correlation between any two trees and the strength of each tree in the forest. Increasing the correlation increases the error rate of the forest. The strength of a tree is determined by the error rate of the tree. Increasing the strength decreases the error rate of the forest. When the forest is growing, random features are selected at random out of all the features in the training data. The best split

on these random features is used to split the node of the tree. The number of random features ($Mtry$) is held constant. Reducing (increasing) $Mtry$ reduces (increases) both the correlation and the strength. The number of features employed in splitting each node for each tree is the primary tuning parameter ($Mtry$). To improve the performance of the random forests algorithm, this parameter should be optimized using training data. The minimum error rate corresponds to the optimal value[4]. Therefore, we use the different values of $Mtry$ to build the forests and evaluate the error rates of the forests. Then, we select the value corresponding to the minimum error rate to build the pattern.

In addition, we need to decide how many trees to grow for each forest ($ntree$), we need $ntree$ to be sufficiently large to ensure a classification error (of a test sample), which is not significantly decreased by adding more trees [62]. According to [58], the overtraining (or overoptimization) vanishes in case of an infinite number of trees.

Oshiro et al [63] suggests a range between 64 and 128 trees in a forest to obtain a good balance between performance, processing time, and memory usage. The same study [63] states that sometimes, a larger number of trees in a forest only increases its computational cost, and has no significant performance gain. Time of execution of the code increases linearly with $ntree$. Larger $ntree$ values lead to slightly more stable values of variable importances, until some values where any further increases having negligible effects [61].

2.5 One Class Support Vector Machine

A one-class support vector machine (one-class SVM) is a popular outlier detection algorithm that was used in various fields such as document and text classification [64][65], machine fault detection [66], intrusion detection system [7][12] and so on. The one-class SVM was firstly proposed by [67] for estimating the support of a high-dimensional distribution. As stated in [67], one-class SVM aims to solve this problem:

"Suppose a data set is drawn from an underlying probability distribution P . Estimate a simple Subset S in the input space such that the probability that a test point drawn from P lies outside of S bounded by some priori specified value between 0 and 1" [7][65].

The solution to this problem is to estimate a function f that is positive on S and negative on the complement.

In other words, in [67], they developed an algorithm which returns a function f that takes the value +1 in a "small" region capturing most of the training data points and -1 elsewhere. Their strategy could be summarized into two steps:

1. Map the data into a feature space corresponding to an appropriate kernel function.
2. Locate a hyper plane that separates the mapped vectors from origin with maximum margin

Let $x_1, x_2, \dots, x_l \in X$ be the training data instances belonging to original space X and l be the number of instances. The one-class SVM may be viewed as a regular binary SVM where all training data lies in the first class and the origin belongs to the second class as shown in Fig 2.2. It finds the maximal margin hyper plane that best separates

the majority of training data -but not all training data to avoid overfitting-, from the origin [7][67].

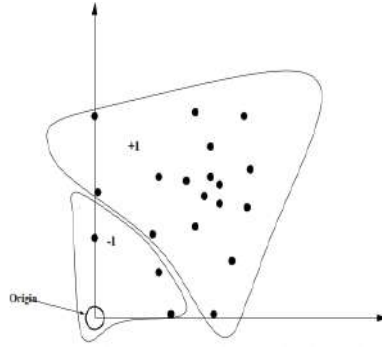


Figure 2.2 one class SVM

Because it is usually difficult to locate a hyper plane that creates training data patterns separable from the origin in the original space X , the SVM uses a feature map $\varphi: X \rightarrow F$, which non-linearly transforms the data from the original space to the feature space in order to locate the hyper plane in the feature space. The one-class SVM also considers a trade-off between maximizing the distance of the hyper plane from the origin and the fraction of data instances contained in the separated region. This is controlled by the parameter ν , which represents the fraction of training instances that can be rejected. The one-class SVM is formulated as the following quadratic program

$$\min_{w, \varepsilon, \rho} \frac{1}{2} \|w\|^2 + \frac{1}{\nu l} \sum_{i=1}^l \varepsilon_i - \rho \quad (18)$$

$$\text{Subject to } (w \cdot \varphi(x_i)) \geq \rho - \varepsilon_i$$

$$\varepsilon_i \geq 0, i = 1, \dots, l$$

Where w is the vector orthogonal to the hyper plane, $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_l$ is the vector of slack variables used to penalize the rejected instances, and ρ represents the margin, i.e. the distance of the hyper plane from the origin.

Because computing in the feature space is difficult due to the curse of dimensionality [66], the SVM utilizes the kernel theory: the inner product in the feature space can be computed using a simple kernel function $k(x, y) = \varphi(x) \cdot \varphi(y)$ such as Gaussian kernel $k(x, y) = e^{-g\|x-y\|^2}$.

By applying the kernel theory and lagrangian multiplier (α_i) to the original quadratic program, the solution of the above equation creates a decision function.

Then, this resulted hyper plane is used to detect outliers of a testing instance by determining to which class the instance belongs. For a generic test instance (z), it is formulated as follows:

$$f(z) = \text{sign} ((w \cdot \varphi(x)) - \rho)$$

The test instance (z) is accepted when $f(z)$ is positive and it is rejected when $f(z)$ is negative. An acceptance indicates that the test instance (z) is considered to be similar to the training data set, and a rejection indicates that it departs from the training data and is considered as an outlier [64][65].

In a one-class SVM model, there are some parameters that affect the characterization of the decision boundary of the training dataset. The parameter ν controls the fraction of training instances that are allowed to be rejected, which means that the decision hyper plane only contains $(1 - \nu) * 100\%$ of training instances. If ν is high, the one-class SVM model only focuses on the most frequent training patterns. In contrast, if ν is very low, the decision hyper plane contains most training instances, including noisy data. The feature mapping also has a significant effect on the decision boundary. The degree of polynomial kernel and the width parameter of the Gaussian kernel control the flexibility of the resulting decision boundary. When using the Gaussian kernel $k(x, y) = e^{-g\|x-y\|^2}$, if the width of parameter g is small, the SVM model loses non-linear power and the decision boundary tends to be smooth. If g is large, the decision boundary of the SVM model tends to be highly sensitive to the training data, which lacks regularization. Therefore, it is important to determine appropriate parameters that consider the classification performance.

When applying the one-class SVM to the anomaly detection, the anomaly detection model is trained using the normal training instances to form the one-class SVM model. Throughout the training procedure, the model locates decision boundaries that separates the normal data from the origin. When it inspects the incoming traffic connections, it detects outliers using the decision function of the model and it classifies the outliers as attack connections [7].

Studies that uses one-class SVM in IDS optimized parameters for ν and g . Default value of g that equals to $1/num_features$ is one heuristic for choosing gamma in various studies; [7] uses different values for parameter $g= 1,0.1,0.01$, and for value of ν it recommended 0.003 or 0.004; [12] used 0.01 for ν parameter.

Chapter 3

Proposed Frameworks

In this chapter we present three contributions; the first is using One-class Classification by Combining Density and Class Probability Estimation for anomaly detection. In the second contribution we propose a new hybrid model using new combinations of classification algorithms. The hybrid model consists of two stages: misuse detection where Hidden Naïve Bayes is used and anomaly detection where one class classification algorithms is used. The last one is a modified version of the proposed hybrid model using k-means clustering algorithm. The details for each model are shown in the rest of this chapter.

3.1 Intrusion Anomaly Detection using One Class Classifier

Many algorithms have been applied to intrusion detection problem as outlier detection algorithms, many examples was stated in chapter 2 for anomaly detection.

Here in this research we use one-class classifier [31] described in chapter 2 as anomaly detection for IDS. OCC is used with Random Forest algorithm for probability estimation and Gaussian density for density estimation. "Normal" is considered as target class, and anomaly/attack class is considered as outlier class. The classifier that is used is random forest. We prefer RF over other choices because of flexibility obtained by it, also RF prove itself as misuse and anomaly detection algorithm for IDS [4]. The system is trained using only normal class instances, artificial data is generated with percentage equals 50% of normal instances to take the role of the second class, i.e. attack class; a training set of normal and artificial data is used to build the OCC model. A test set with normal and attack "outlier" instances will be evaluated using the built model.

3.2 Intrusion Detection using Hybrid Model

Our proposed hybrid framework combines the misuse detection component and the anomaly detection component to overcome the limitations for both methods. Figure 3.1 shows block diagram for the proposed framework. A detailed explain of the misuse and anomaly detection components are shown in Figure 3.2. There are two phases in the framework: misuse phase and anomaly phase. The system can build patterns of intrusions for the misuse detection component to detect known intrusions and can detect unknown intrusions using the anomaly detection component. **In the misuse phase**, after preprocessing the training data, the preprocessed data are stored in the training database. The intrusion pattern builder module is trained from the data in the training database, and it outputs patterns of intrusions to the misuse detector module. In the testing, network traffic is captured by the network sensors and fed to the misuse detector after being preprocessed. The misuse detector raises an alarm to the misuse alarmer if any connection matches an intrusion pattern. Then, the misuse alarmer delivers alarms to security officer. If the connection does not match any intrusion pattern, it will sent to the anomaly database. **In the anomaly phase**; first, the anomaly pattern builder module retrieves data from the training anomaly database to build patterns of normal network connections, and outputs the built patterns to the

outlier detector module. The outlier detector retrieves the data from the anomaly database and uses the outlier detection technique to detect novel attacks. If it detects any attack, it raises alarms. It can also store the newly detected intrusions in the training database so that the new intrusion patterns can be built for misuse detection.

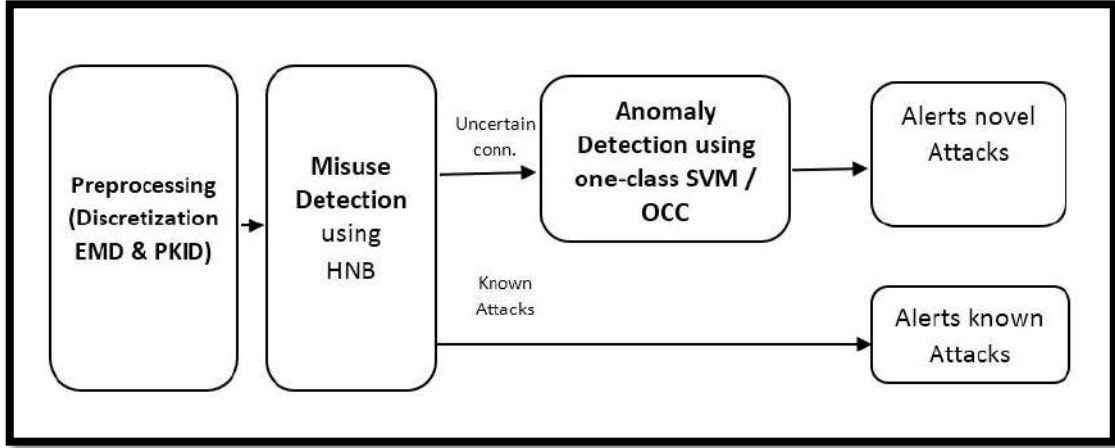


Figure 3.1 Block diagram of hybrid model

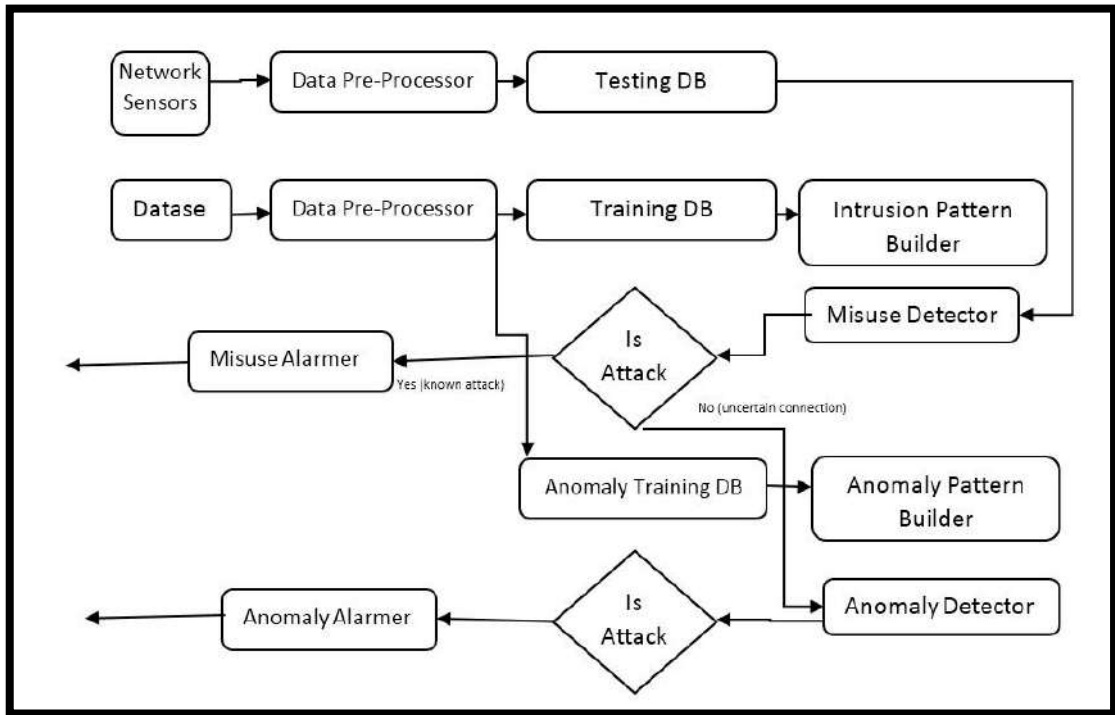


Figure 3.2 Proposed hybrid model

The performance of the anomaly detection tends to be reduced with increasing number of attack connections [4]; we overcome this problem by removing the known attacks through the misuse detection first, the number of attacks can be reduced significantly in the datasets fed to the anomaly detection phase.

First stage applies Hidden Naïve Bayes algorithm (HNB) as misuse detection algorithm, preprocessing of dataset is needed to build HNB model so discretization with two type of discretization methods PKID and EMD are used. Then anomaly detection stage applies one class classification algorithms on normally classified instances from the output of the first stage to detect outliers (anomalies) that does not detected by misuse stage. Training and testing processes are described in Figure 3.3 and 3.4, respectively.

At training step, dataset (Normal and Anomaly Instances) used to build profiles (build model) for first stage – misuse detection- using HNB, and only normal instances of training dataset is used to train anomaly detector (one class classifier or one class SVM). For testing step; a test set will fed into the framework after discretization process, HNB model will classify test set as built model and only instances that has been classified as normal will allowed to pass to anomaly detection model.

For anomaly detection we take into account test points that classified as normal (contains truly normal instances and anomaly instances that incorrectly classified as normal); other values that classified as anomaly (contains truly anomaly instances and normal instances that incorrectly classified as anomaly) are not considered. The reason for this choice is that:

- ✚ The total number of the normal instances that classified as anomaly resulted from misuse detection is small and very small compared to normal data so that DR of normal data is very high.
- ✚ The total of the anomaly instances that classified as normal resulted from misuse detection is large compared to anomaly data, this because of novel attacks that does not appears in the training dataset. This type of data is severe and dangerous. It is a big problem for the system/network to classify attack connection as normal connection.
- ✚ Profiles normal instances is the better choice (vs. profiles attack instances) since usually no novel normal instances found in testing set; but on the other side we have novel attacks that is not shown by the system and does not learned in training step.

Now for anomaly detection stage in our proposed framework we will work with two algorithms, one class classifier with random forest and one class SVM, each one is used to build patterns over NSL-KDD and 10% KDDCup datasets discretized using EMD and PKID. Also we built models for anomaly detection using these algorithm for whole data without using misuse detection step and compare it with the proposed hybrid model.

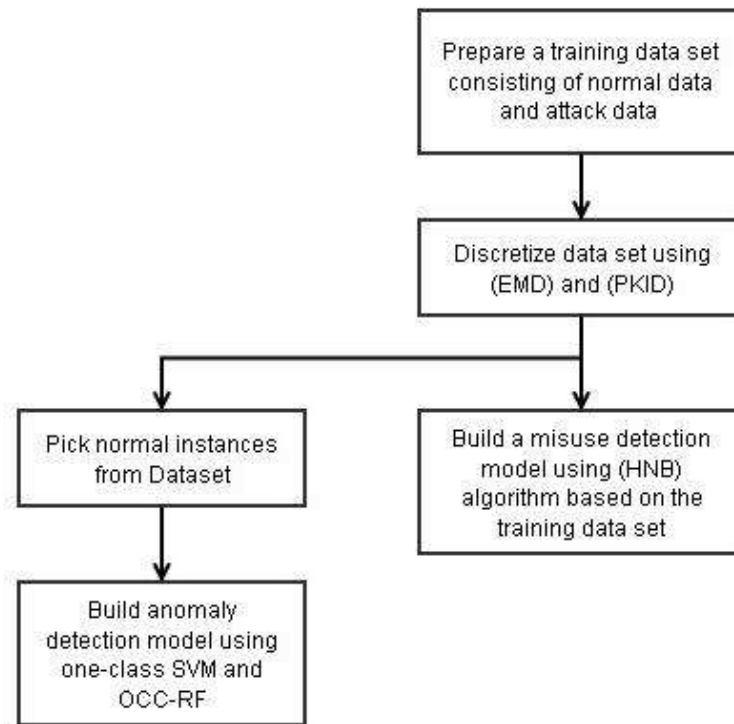


Figure 3.3 Training process of the proposed hybrid intrusion detection method

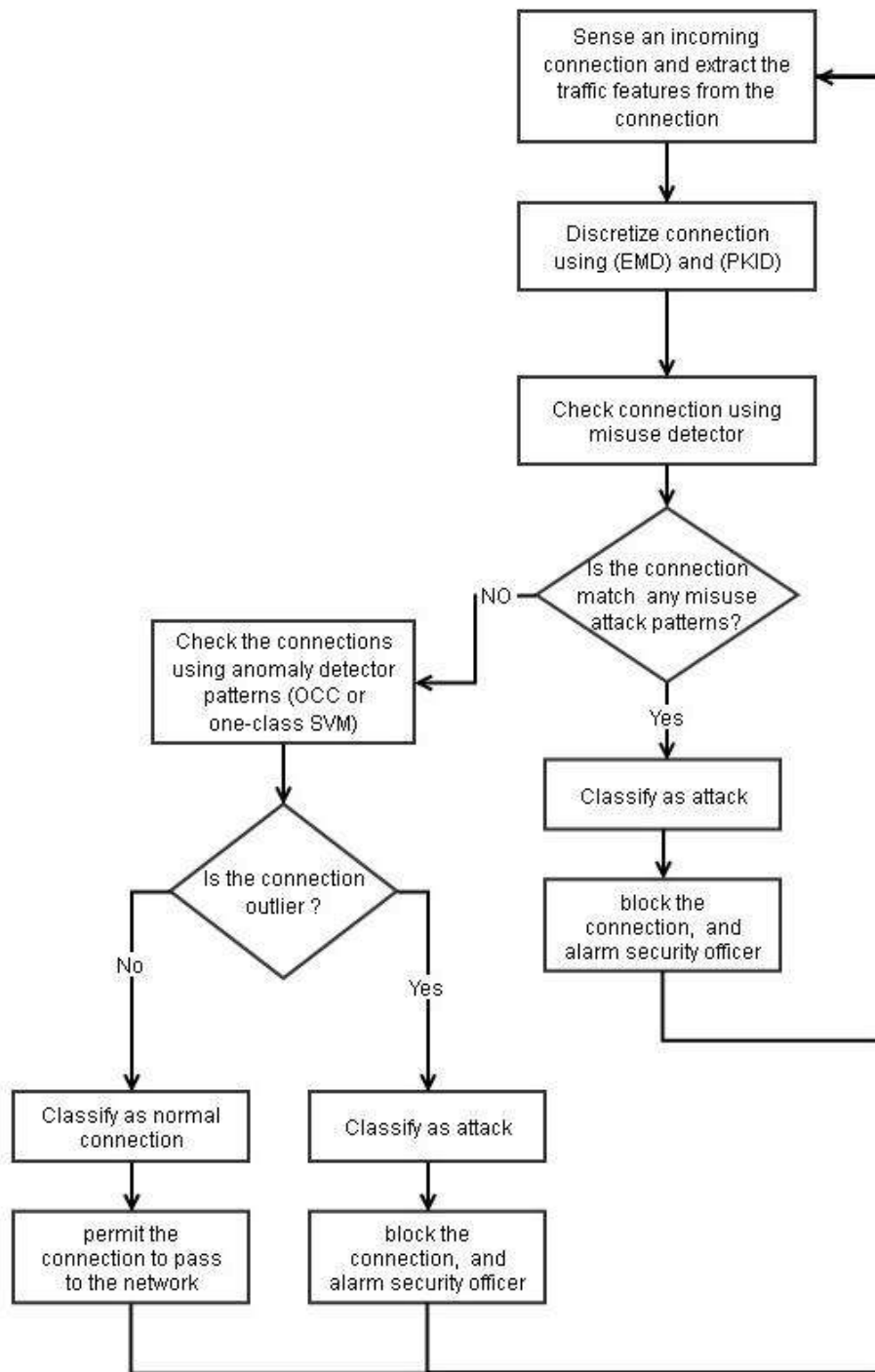


Figure 3.4 Testing process of the proposed hybrid intrusion detection method

3.3 Improved Hybrid Model Using Clustering

The proposed improved hybrid framework is similar to hybrid framework presented in the previous section in misuse detection phase, the difference is the using of clustering algorithm in anomaly detection phase. Figure 3.5 is a block diagram for the improved hybrid model using clustering. After using HNB for misuse detection, clustering algorithm is used to decompose uncertain connections –connections classified as normal by misuse detector- to k clusters, and finally outlier model is trained for each normal training subset corresponding to a cluster i ($1 < i < k$).

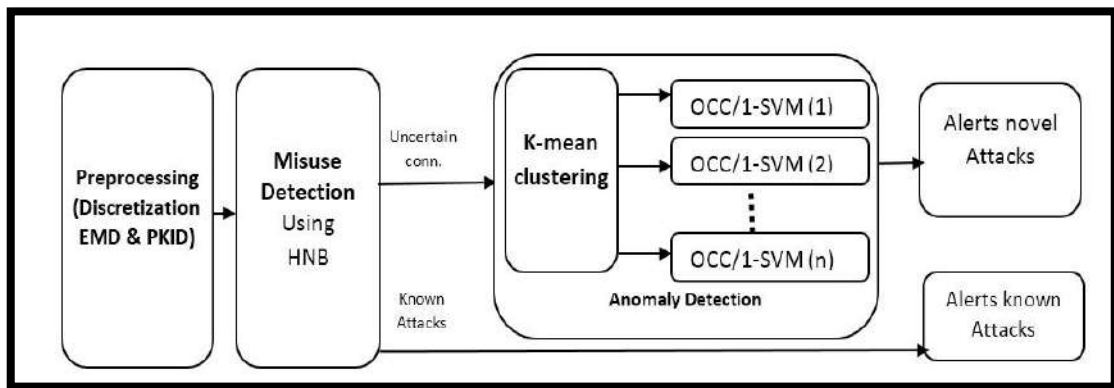


Figure 3.5 Block diagram for improved hybrid model using clustering

Training and testing step is similar to hybrid model; where at training step, whole points in the dataset (Normal and Anomaly instances) are used to build patterns for misuse detection phase. While at anomaly detection phase, only normal instances are clustered using k-means clustering algorithm to produce k clusters, the k resultant clusters are used to build k models, using either one class classifier or one-class SVM algorithms, each model is uses only normal instances of the corresponding cluster. Because each cluster will focus on its own instances, this will make building model easy and fast. For testing step; a test set will fed into the framework after discretization process, HNB model will evaluate a test instance using the built model and only instances that has been classified as normal will clustered to the most similar cluster; finally classification of each instance will performed according to outlier model corresponds to incoming cluster. Figures 3.6 and 3.7 show the training and testing processes, respectively.

The primary reason for decomposing the normal training data set is that the anomaly detection models in the previous hybrid intrusion detection methods have attempted to profile the normal connection patterns using one outlier detection mode; however, in reality, there are various patterns according to the protocol type (TCP, UDP, ICMP, etc.), service type (HTTP, FTP, SNMP, etc.), and so on.

The decision boundaries of the anomaly detection model in the clustering model can describe the normal behavior better than those in hybrid model and conventional methods; as each model is focus on the normal data in the smaller region because each model of the decomposed region profiles only its corresponding normal patterns; so it

is expected that the clustering model can detect attacks with a lower false positive rate compared to hybrid model.

Clustering of uncertain data to k clusters alleviates the sensitivity of parameters to the training data set, because the data patterns of each decomposed subset are less complex than those of the whole data set, multiple models for each decomposed data pattern can be less flexible than a single model for the whole data pattern.

Why using k-means for clustering?

K-means is fast, simple and robust which allows it to run on large datasets [68]. We use K-means despite its limitation discussed in many studies includes [68]; k-means is not robust to noisy data and outliers, we neglect this as we use it to decompose data to k clusters based on similarity between instances (K-means minimize intra-class variance, the sum of squared distances from each data point being clustered to its closest cluster center), and outlier detection is the job of one class classifier algorithm which comes next after clustering step. We run k-means with k equals ten and with k-means ++ initialization method [69] to overcome the randomly initializing centroids of clusters.

For note we use $k = 10$ for k-means, so to get clusters ; this value is used for test and may higher or lower values is more optimum and more efficient with some datasets and gives more homogeneous clusters from the side of splitting normal form anomaly connections is different clusters.

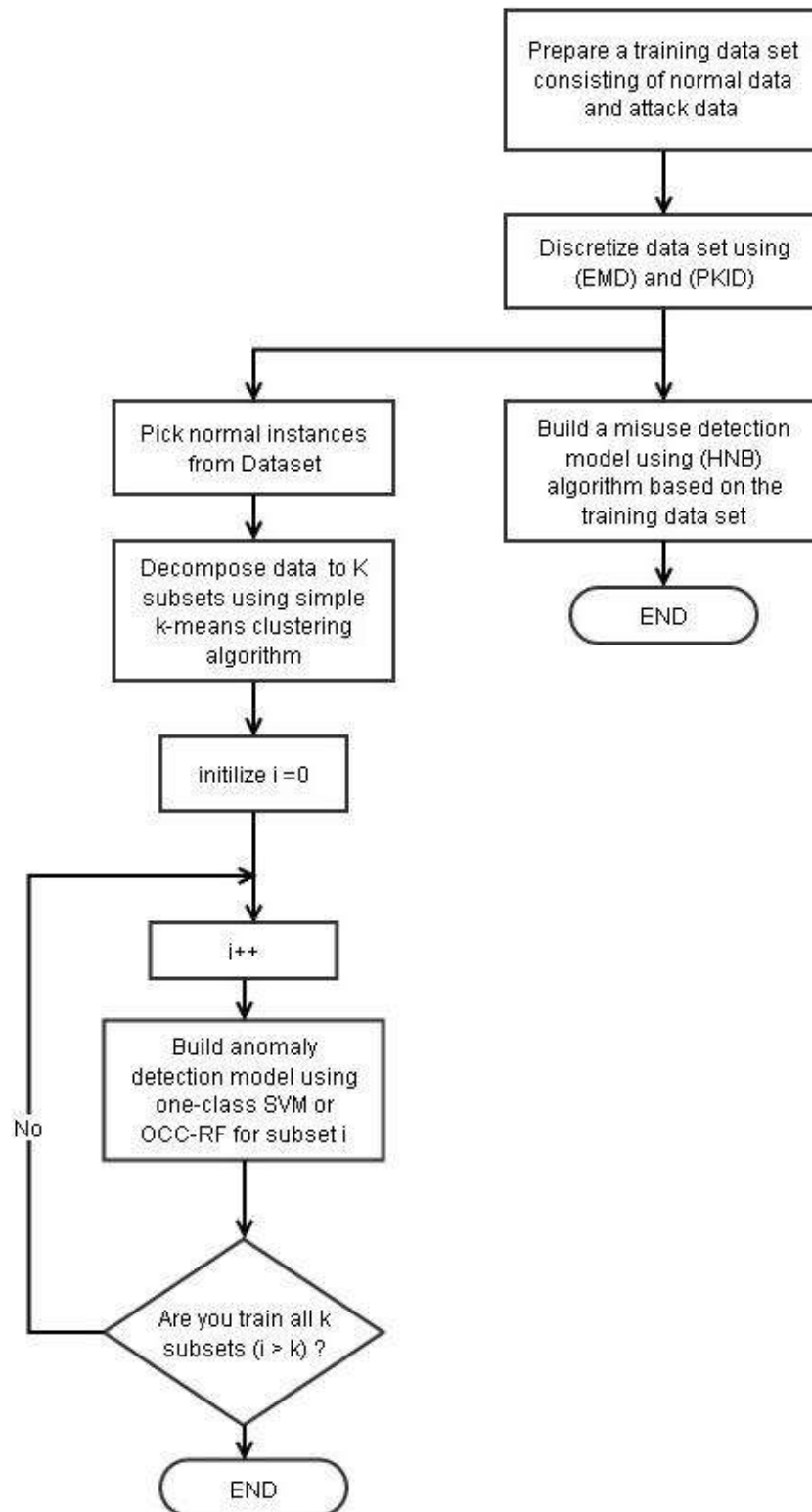


Figure 3.6 Training process of the proposed improved hybrid intrusion detection method

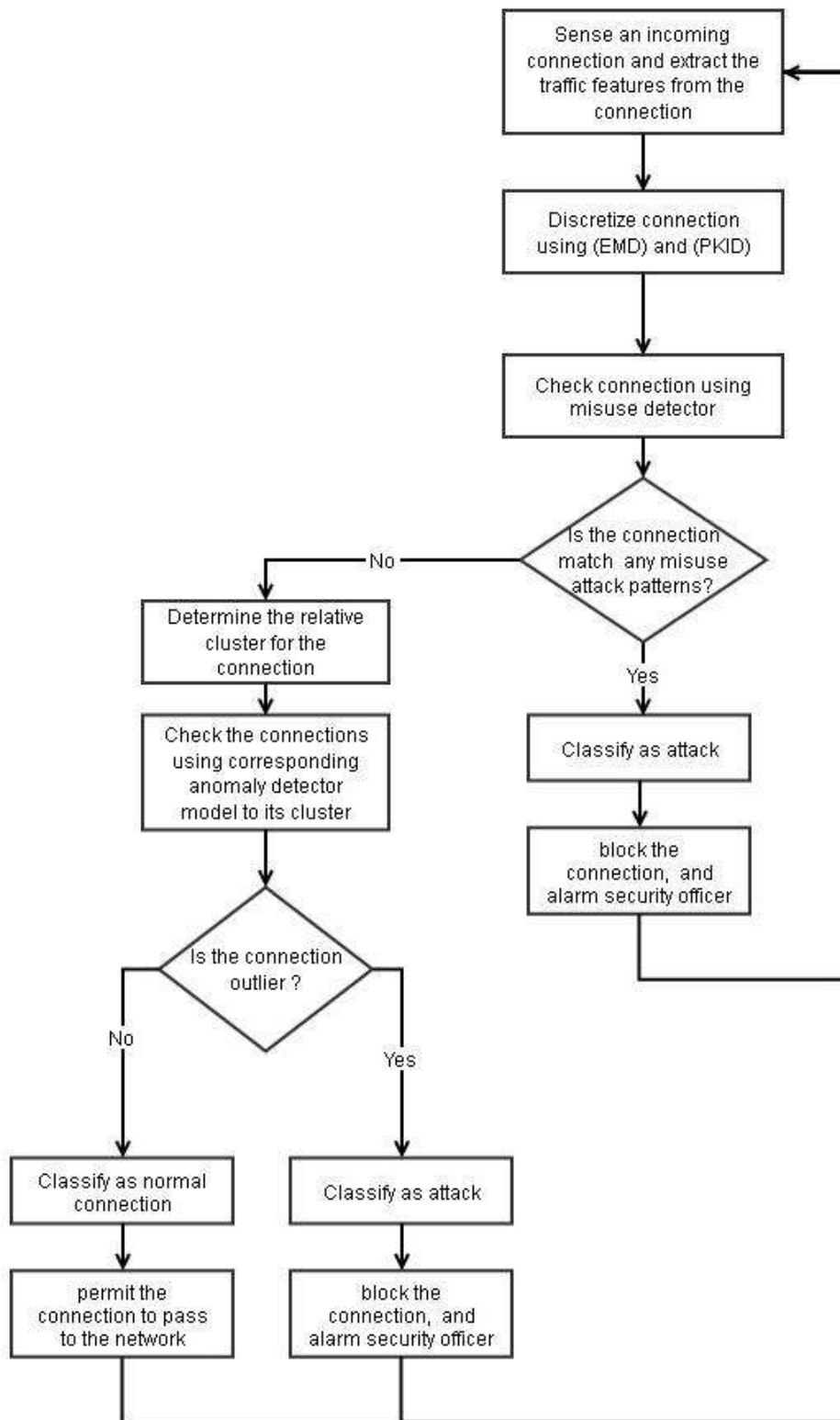


Figure 3.7 Testing process of the proposed improved hybrid intrusion detection method

Chapter 4

Experimental Results

In this chapter we will display the results of each contribution mentioned in chapter 3, we will show the parameter optimization for each model and discuss the obtained results.

4.1 Dataset

Currently, there are only few public datasets for network-based IDSs like KDDCup'99, the majority of the experiments in the intrusion detection domain performed on these datasets [36].

Datasets that are used for intrusion detection are categorized in three categories DARPA, KDDCup and some real world datasets. The KDDCup dataset is widely used by the researchers to test the effectiveness of the developed method for intrusion detection with 42%, 20 % of the studied papers used DARPA dataset to check the effectiveness of the methods for intrusion detection, and the rest of the studied papers used other real world data sets [1].

In our research KDDCup'99¹ dataset will be used. We believe it still can be applied as an effective benchmark dataset to help researchers compare different intrusion detection methods.

KDDCUP dataset contains training data that include seven weeks of network traffic in the form of TCP dump data consisting of approximately 5 million connection records, each of which is approximately 100 bytes. The test data included two weeks of traffic, with approximately 2 million connection records [36].

The training data contain 24 attack types, and the test data contain 38 types, all of which are mapped to four basic attack classes as shown in Table 4.1, the basic attack classes are Denial of Service (DOS), Remote to Local (R2L), User to Root (U2R), and Probing attack [21].

Table 4.1 Mapping attack types to the attack classes on KDDCup 1999 dataset [36]

Class	Attacks in the training data	Additional Attacks in the testing data
DOS	Back, land, Neptune, pod, smurf, teardrop	apache2, mailbomb, processtable, udpstorm
Probe	ipsweep, portsweep, satan, nmap	mscan, saint
U2R	buffer_overflow, loadmodule, rootkit, perl	httptunnel, ps, worm, xterm
R2L	ftp_write, guess_passwd, imap, multihop, warezmaster, warezclient, spy, phf	named, sendmail, snmpgetattak, snmpguess, sqlattack, xlock, xsnoop

¹ Dataset KDDCup is available on web site <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>; different version is available for supervised and unsupervised learning, also reduced version 10% KDDCup is available which is used in our study.

Table 4.2 Characteristics of KDDCup 1999 dataset (10% KDDCup version)

class	Training data	Testing data
Normal	19.69	19.48
Probe	0.83	1.34
DOS	79.24	73.90
U2R	0.01	0.07
R2L	0.23	5.20

Table 4.2 shows the distribution of the classes of 10% KDDCup dataset, a reduced version of KDDCup dataset. As shown in Table 2, the distributions of the classes are not necessarily the same in the training and test datasets. Only approximately 20% of the records are categorized as normal connections.

Each connection record contains 7 discrete and 34 continuous features for a total of 41 features. Each record captures various connection features, such as service type, protocol type and the number of failed login attempts.

In addition to the possible inter-dependence between some features, the high data dimensionality of the dataset due to its large feature set poses a significant challenge to any data mining model. The dataset's continuous features also result in difficulties for many data mining models [36] including HNB and other Naïve Bayes models. Discretization is commonly used to convert continuous features into their discrete counterparts. Furthermore, discretization improves the performance of classifier models on large datasets [48], including the KDD'99 dataset [52].

NSL-KDD² is a modified version of KDDCup'99, which recently used in some studies. NSL-KDD is a dataset suggested to solve some but not all of the inherent problems of the KDDCup'99 dataset [21]. The NSL-KDD data set has the following advantages over the original KDD data set:

- ✚ It does not include a number of redundant records in the train set and test set which cause the learning algorithms to be biased towards the frequent records, and thus prevent them from learning unfrequent records which are usually more harmful to networks such as U2R and R2L attacks. In addition, the existence of these repeated records in the test set will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records.

² Dataset NSL-KDD is available on web site <http://nsl.cs.unb.ca/NSL-KDD>; attack labeled version is used in our study.

- ✚ The number of records in the train and test sets are reasonable with comparing to full KDDCup data set. And this make it affordable to use without needing for sub-sampling and randomly selection data sets.
- ✚ The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDDCup data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.

We use the labeled 10%KDDCup dataset and the improved version NSL-KDD in our study, we use training dataset to build the models and test dataset to evaluate the built models. 10%KDDCup training dataset consists of 494021 training instances and about 311029 testing instances, while NSL-KDD contains 125973 training instances and 22544 test instances.

We use 10%KDDCup without randomly selection or subsampling as used in many studies [3][4][12]; and we use NSL-KDD dataset, despite its improvements for KDDCup dataset, it stills balanced and considered small image for full KDDCup dataset which does not require resampling.

4.2 Software and Tools

In our experiments we use Weka tool³ [70] (Waikato Environment for Knowledge Analysis) and LibSVM library⁴ to evaluate our proposed methods. Weka is an open source software written in java, it has a collection of machine learning algorithms for data mining tasks and is widely used for teaching and researching. LIBSVM is an integrated software for support vector classification and regression which can handle one-class SVM. Experiments runs on machine with core i5 processor and 8 GB memory.

4.3 Evaluation Measurement:

Regarding to the previous researches in IDS area, the performance of IDS is measured and evaluated by calculated confusion matrix, Table 4.3 shows the components of the confusion matrix.

Table 4.3 Confusion Matrix

		Predicted Class	
		Negative Class (Normal)	Positive Class (Attack)
Actual Class	Negative Class (Normal)	True Negative (TN)	False Positive (FP)
	Positive Class (Attack)	False Negative (FN)	True Positive (TP)

³ Weka software and source code is available to download from <http://www.cs.waikato.ac.nz/~ml/weka/> website for both x32 and x64 bit computers

⁴ LIBSVM provides a simple interface where users can easily link it with their own programs. Java and C++ source is available and many interfaces and extensions including Weka interface are available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

- ✚ True positive indicates the number of attack records that are correctly classified, true positive be a sign of properly detecting the occurrence of attack in IDS.
- ✚ True negative indicates the number of normal records that are correctly classified, true negative specifies that the IDS have not made a mistake in detecting a normal condition.
- ✚ False positive indicates records that were incorrectly classified as attacks whereas in fact they are valid activities. A false positive specifies the wrong detection of a particular attack by IDS. A false positive is often produced due to lose recognition conditions and it represents the accuracy of detection system.
- ✚ False negative indicates records that were incorrectly classified as normal activities whereas in fact they are attacks. A false negative indicate that the IDS is unable to detect the intrusion after particular attack has occurred.

From the confusion matrix, different values can be obtained:

- ✚ True Negative Rate (specificity) = $TN/(TN+FP)$
- ✚ True Positive Rate (Detection Rate) (Sensitivity)(Recall) = $TP/(TP+FN)$
- ✚ False Positive Rate = $1-\text{specificity} = FP/(TN+FP)$
- ✚ False Negative Rate = $FN/(TP+FN)$

Like most studies in the field of IDS, we use detection rate and false positive rate as evaluation metric for our experiments. We use receiver operating characteristic curve (ROC) to show the performance of our proposed models and compare it to others.

Receiver operating characteristic (ROC) curve, is a graphical plot that illustrates the performance of a classifier system as its discrimination threshold is varied as in our case when using outlier/anomaly detection algorithms. The curve is created by plotting the true positive rate against the false positive rate at various threshold settings. (The true positive rate is also known as detection rate, sensitivity, or recall in machine learning. The false-positive rate is also known as the fall-out and can be calculated as $(1-\text{specificity})$). The ROC curve is thus the sensitivity as a function of fall-out. ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from the cost context or the class distribution.

4.4 Data Preparation

KDDCup and NSL-KDD are multiclass datasets; the original datasets files downloaded from website labels the data by attack type (i.e. normal, nepton, snmpattack, and so on). We divide connections to five classes according to Table 1, in addition to normal class. Experiments by [57] states that “If no novel classes are expected after the classifier has been trained one should use multi-class classification without relabeling; and if novel classes are expected, then the training data should be relabeled to ‘target’ and ‘outlier’, where the former is the single class we are attempting to verify, and the latter contains all other classes relabeled to a single combined class. If there is a limited number of non-target classes, or they do not sufficiently cover

possible novel cases for some other reason –as in our case-, then one should use one-class classification. Otherwise, one should use two-class classification”. Therefore we relabel all attacks classes to “anomaly” for training and testing sets in order to use one-class classification.

After relabeling dataset we use Weka supervised discretization filter “EMD” and unsupervised discretization filters “PKID” to produce the discretized datasets.

4.5 Results of Intrusion Anomaly Detection using One Class Classifier

This sections shows and discusses the results obtained using OCC-RF as anomaly intrusion detection algorithm.

4.5.1 Parameter Optimization

We build OCC model with gaussian density as density estimator and random forest algorithm as probability estimator, percentage of generating artificial data is set to 50% of the size of the target class, and the target class is “normal” class.

We control the model building using random forest parameters, $Mtry$ and $ntree$. Different values of number of the random features ($Mtry$)(1,2,6,10,20) are used over 10%KDDCup dataset once and over NSL-KDD dataset, each value with different number of trees ($ntree$) in the forest varying from 2,5,10,20,50,100,200. We plot ROC curve for different values of $Mtry$ for both datasets.

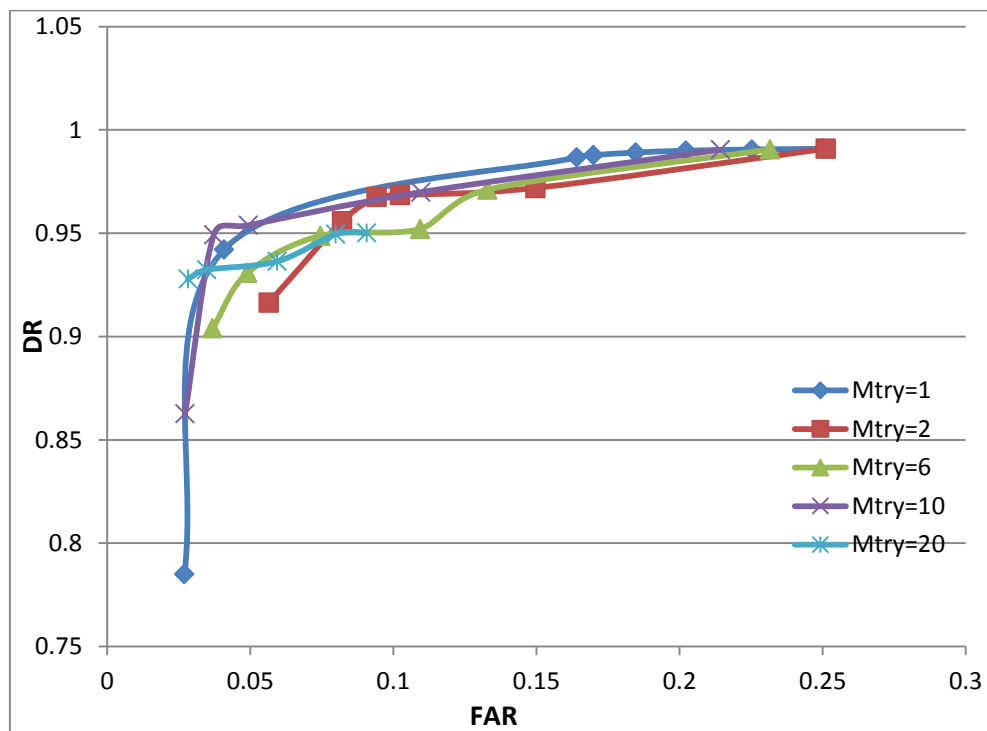


Figure 4.1 ROC Curve For 10%KDDCup with EMD obtained by OCC-RF

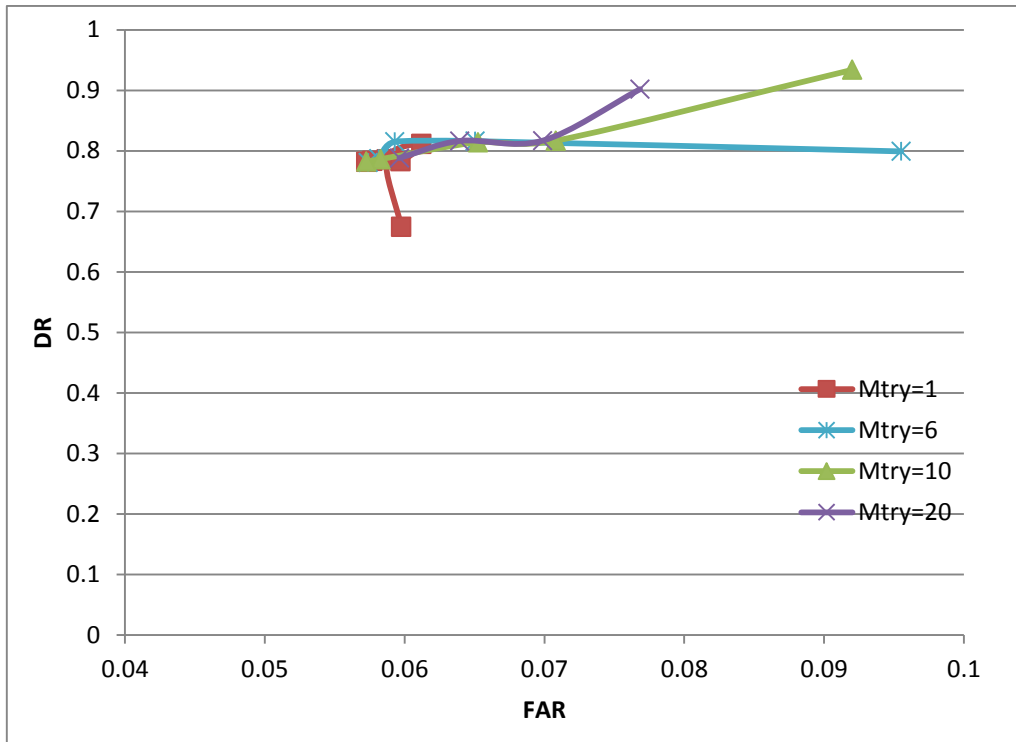


Figure 4.2 ROC Curve For 10%KDDCup with PKID obtained by OCC-RF

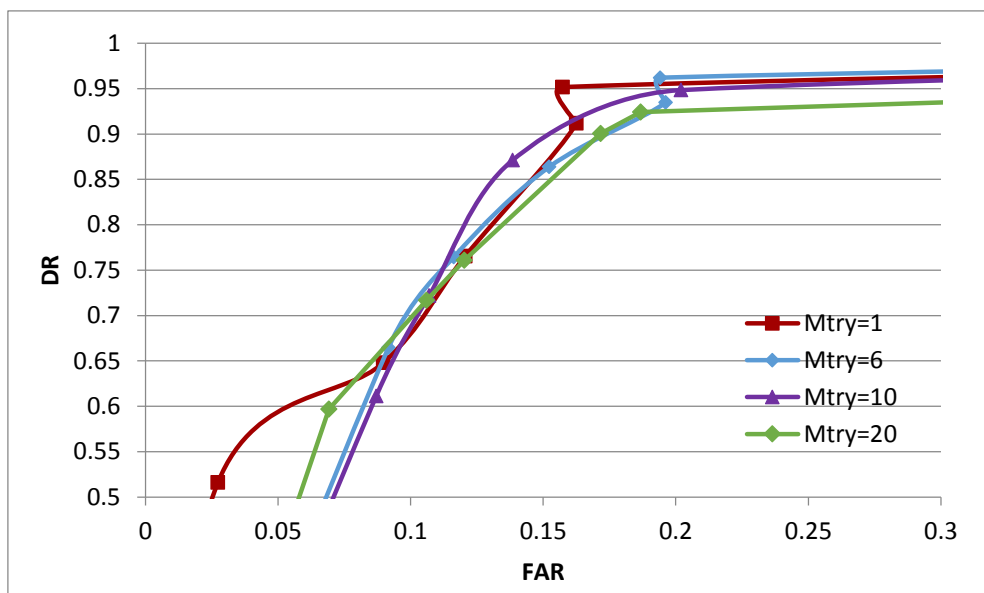


Figure 4.3 ROC Curve for NSL-KDDCup with EMD obtained by OCC-RF

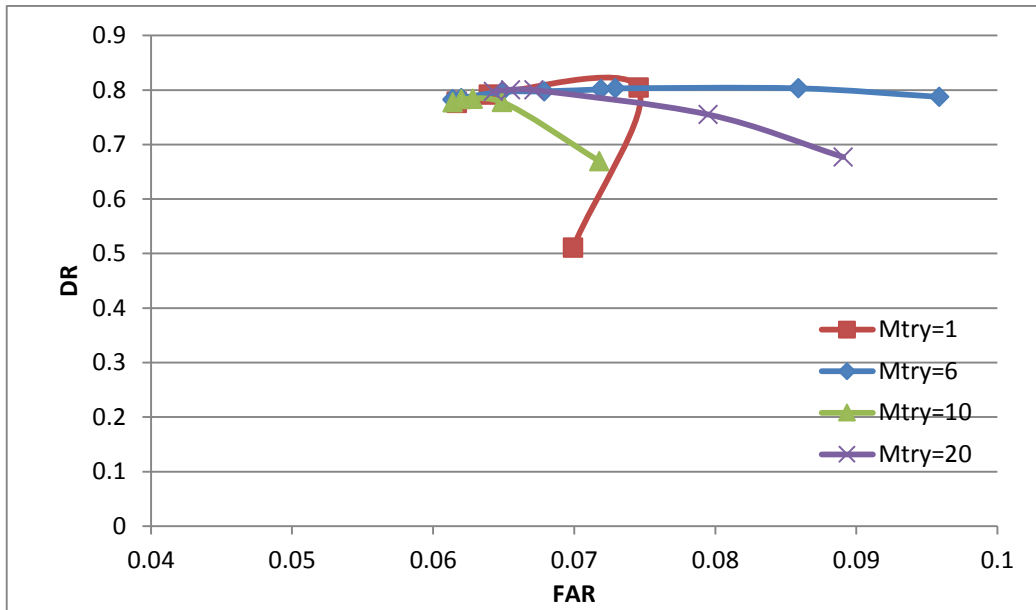


Figure 4.4 ROC Curve for NSL-KDDCup with PKID obtained by OCC-RF

4.5.2 Evaluation and Discussion

NSL-KDD and 10% KDDCup datasets discretize using EMD:

Figure 4.1 presents the ROC curves of the attacks for 10%KDDCup dataset that obtained when using OCC-RF as anomaly detection, with variations in parameters $Mtry$ and $ntree$ as we mentioned earlier. From the ROC curves we note that OCC-RF anomaly model can reach over 0.95 detection rate value with FAR does not exceed 0.05 which is desirable, and reaches 0.989 with FAR 0.18.

Figure 4.3 presents the ROC curves of the attacks for NSL-KDDCup dataset that obtained when using OCC-RF as anomaly detection, with variations in parameters $Mtry$ and $ntree$ as we mentioned earlier. From the ROC curves we note that OCC-RF anomaly model can reach 0.93 detection rate with FAR equals 0.16.

For small number of trees, FAR is small and acceptable range of DR is achieved; as we increase number of trees DR becomes higher with increasing in FAR value. This is shown by Figures 4.1 and 4.3. The same behavior results from all models build with different values of $Mtry$ parameter.

When studying the effect of changing $Mtry$, we show that increasing $Mtry$ does not always improve the performance as we show in Figure 4.1 that when $Mtry$ equals 1 gives optimum solutions where $Mtry = 6$ (the default value) is lower performance than $Mtry = 1$, and $Mtry = 10$ is better than $Mtry = 6$. This is related to nature of data (relations between points) and pre-processing step EMD and PKID discretization.

Besides, increasing $Mtry$ increases the time to build the pattern, we notice that higher values of $Mtry$ over 20 predominantly does not add improvement in DR and FAR but add time overhead to build the model. So values lower than 20

are used in our experiments. Optimum values is achieved when $Mtry = 1$ for both datasets as shown in Figures 4.1 (diamond mark curve) and Figure 4.3 (rectangular mark curve).

✚ NSL-KDD and 10% KDDCup datasets discretize using PKID:

Figures 4.2 and 4.4 show the ROC curve for 10%KDDCup and NSL-KDD using PKID discretization, respectively. Figures shows that most resultant values lies between 0.06 and 0.08 error rate and DR around 0.8 for both 10%KDDCup and NSL-KDD.

As shown in Figure 4.2 and 4.4 FAR for both datasets begins higher at lower number of trees parameter values and decreases with increasing number of trees parameter values until reach stable point. In contrast for 10%KDDCup dataset DR begins higher relatively – often not always- then go lower and stay oscillating in the same range with small increase or decrease- while In NSL-KDD DR begins lower – relatively- then go higher until reach stable point.

Although OCC with PKID discretization starts with unstable steps (higher or lower DR with higher FAR) it finally reaches local minima and vibrate around it with small changes –stable value for many steps after.

Figures 4.5 and 4.6 show the ROC curves for using OCC-RF and one-class SVM as anomaly detection algorithms for NSL-KDD dataset. Optimal values is used in this comparison for both models. **For EMD data**, for smaller values of DR one-class SVM model is better than OCC-RF model. The result changes to the side of OCC-RF model for higher values of DR. **For PKID data**, most of the values obtained by OCC-RF model are concentrating in a small region, as we show that values are between 0.06 and 0.08 FAR while DR is about 0.80. Rather than OCC-RF fails to reach higher detection rates the values that reaches are acceptable and better than one-class SVM anomaly model.

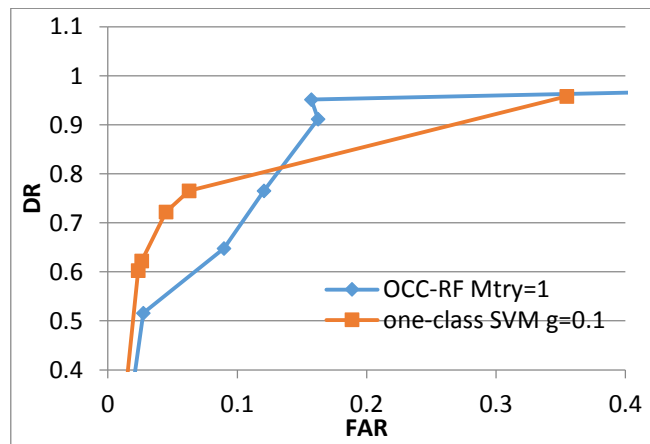


Figure 4.5 ROC curves of OCC-RF and one-class SVM anomaly detection models for NSL-KDD – EMD dataset

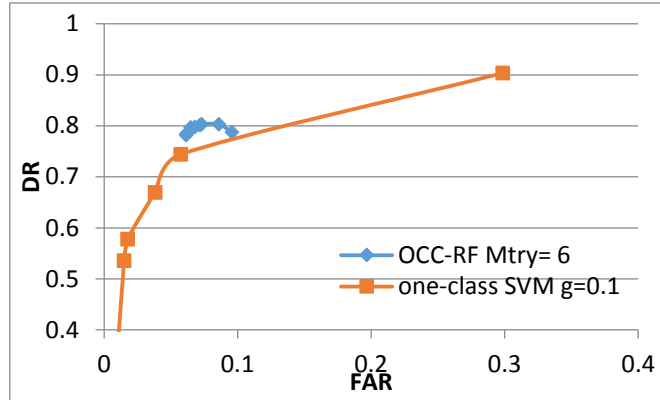


Figure 4.6 ROC curves of OCC-RF and one-class SVM anomaly detection models for NSL-KDD – PKID dataset

4.6 Results of Intrusion Detection using Hybrid Model

This section shows and discusses the results of misuse detection using HNB algorithm, following by the results of hybrid model used HNB as anomaly detection with one class classifier (OCC-RF) and one-class SVM as outlier detection algorithms; the last subsection show fast comparison between the two proposed hybrid models.

4.6.1 Results of misuse detection

As stated in [36] study, Hidden Naïve Bayes classification model augmented with various discretization and feature selection methods exhibits better overall results than the traditional Naïve Bayes model, and the leading extended Naïve Bayes models, in terms of detection accuracy and error rate.

Because of its simplicity and its advantage over the extended Naïve Bayes model's conditional independence assumption, Hidden Naïve Bayes is convenient for datasets with dependent attributes like KDDCup'99 intrusion detection dataset. As we introduce we use Hidden Naïve Bayes algorithm for misuse detection. We use equations mentioned in section 2.3 to calculate classifier prediction. Joint distribution estimation, calculation of hidden parent and corresponding weights is according to these equations.

In misuse detection, we build the patterns of normal and intrusions instances. With the built patterns, we use the misuse approach to classify test set (detecting intrusions over the test set). The detection rate and false positive rate for NSL-KDDCup and 10%KDDCup for both EMD and PKID discretization are shown in Table 4.5.

Table 4.5 Results of misuse detection using HNB algorithm

	DR	FAR
NSL-KDD-EMD	0.574	0.028
NSL-KDD-PKID	0.588	0.061
KDDCup-EMD	0.906	0.014
KDDCup-PKID	0.909	0.014

From Table 4.5 we conclude to there is no difference between behavior of HNB as misuse detection algorithm between data discretize using EMD or PKID algorithm for 10% KDDCup dataset and test set; on other hand data discretized using PKID convergent result with EMD but with higher FAR for NSL-KDD dataset. The noticeable difference in DR between NSL-KDD and KDDCup datasets is refer to number of novel attack data with respect to known attacks in each dataset.

Now we expect that anomaly detection phase will increase the DR also FAR but DR starts where misuse ends. As an example if we have 1000 instance (700 normal) and HNB detect 620 of normal and detect 200 of attack, the rest 100 attack instances classified as normal. Then anomaly detection system detect 120 outlier (60 and 60) then DR will increase and this is what we want, and FAR is also increase and that is rejected.

This because each phase is deal with different partition of data; while misuse detection is take care of know attacks and only novel attacks/unknown patterns are remained for outlier detection algorithm, as a result DR will increase; similarity between normal and some attack cause FAR to increase also. The same reason cause anomaly detection to detect some types of attacks well and other types of attacks that not able to detect. The attacks that failed to detect in the same region of normal data in the used feature space in one-class SVM model, or fall into the same leaf of a tree built by the random forests algorithm in OCC-RF.

For NSL-KDD the playing court for anomaly detection is wide because HND will only reach 0.57 DR with approximately 0.03 FAR; but in 10%KDDCup dataset HND will reach over 0.9 DR with lower than 0.015 FAR so the playing court is narrow, as shown in Table 4.5.

4.6.2 Hybrid Framework with random forest one-class classifier

This section shows the result for hybrid model used HNB as misuse detection with one class classifier (OCC-RF) as outlier detection algorithm.

At training step; after building HNB model, normal instances are used to build patterns for anomaly detection using OCC-RF. HNB model used to evaluate a test set, then instances that classified as normal are fed into the OCC-RF anomaly detection model to determine outlier. We compare our proposed hybrid framework HNB-OCC with OCC-RF anomaly detection model, again normal class instances are used to learn the system and built patterns and test set is evaluated over the built patterns.

We build OCC model with gaussian density as density estimator and random forest algorithm as probability estimator, percentage of generating artificial data is set to 50% of the size of the target class, and the target class is “normal” class. We optimize parameters for classifier algorithm -random forest- used as probability estimation for OCC. Parameters ($Mtry$ and the number of trees) are optimized. We build patterns of normal data using $Mtry$ (1, 6($default\ value = int(log_2(4l+1))$), 10, 20) for different ntree values (5, 10, 20, 50, 100 and 200) and detect outliers (novel attacks often) from the anomaly test set using the built patterns.

Figures 4.7- 4.10 show the results for the proposed hybrid model HNB-OCC applied to different dataset combinations, 10%KDDCup_EMD, NSL-KDD_EMD, 10%KDDCup_PKID, and NSL-KDD_PKID respectively. Each figure contains sub-figures presents the ROC curves of the attacks for the proposed hybrid model HNB-OCC (shown in red color) and its comparison anomaly detection model using OCC-RF (shown in blue line) with different values of $Mtry$ parameters as discussed before; the last sub-figure plots ROC curves for different values of $Mtry$ parameter for the proposed hybrid model HNB-OCC. The last figure, Figure 4.10 shows only ROC curves when $Mtry = 1$; where other values of ROC curves is not comparable in this case.

Figures 4.7- 4.10 show that our hybrid framework HNB-OCC outperforms or similar performance of anomaly detection model using OCC-RF algorithm.

For EMD data; Figure 4.7 and 4.8 show that for smaller values of n_{tree} hybrid algorithm outperforms anomaly detection using OCC, then the behavior is convergent as n_{tree} value increases.

For PKID data; the improvement in DR for the proposed hybrid model over anomaly model at 10%KDDCup dataset is mostly above 10% better as shown in Figure 4.9; but for NSL-KDD the values we can compare shows improvement with only 5%, as shown in Figure 4.10.

We optimize random forest parameters $Mtry$ and n_{tree} to reach best model for OCC-RF; optimum values will be when $Mtry$ 1 and 10 for EMD models as shown in Table 4.6 and Figures 4.7 and 4.8; from time perspective lower $Mtry$ values is faster than, so $Mtry = 1$ is better. On the other hand $Mtry$ equals 10 may be consider the optimum value for PKID for 10%KDDCup dataset and $Mtry= 20$ for NSL-KDD dataset.

Table 4.6 Optimal Value for hybrid model using OCC-RF

10%KDDCup-EMD	$Mtry= 1$ and 10	Figure 4.7 (rectangular and diamond mark curves)
10%KDDCup-PKID	$Mtry= 10$	Figure 4.9 (x mark curve)
NSL-KDD-EMD	$Mtry= 1$ and 10	Figure 4.8 (diamond and triangular curves)
NSL-KDD-PKID	$Mtry= 20$	Figure 4.10 (x mark curve)

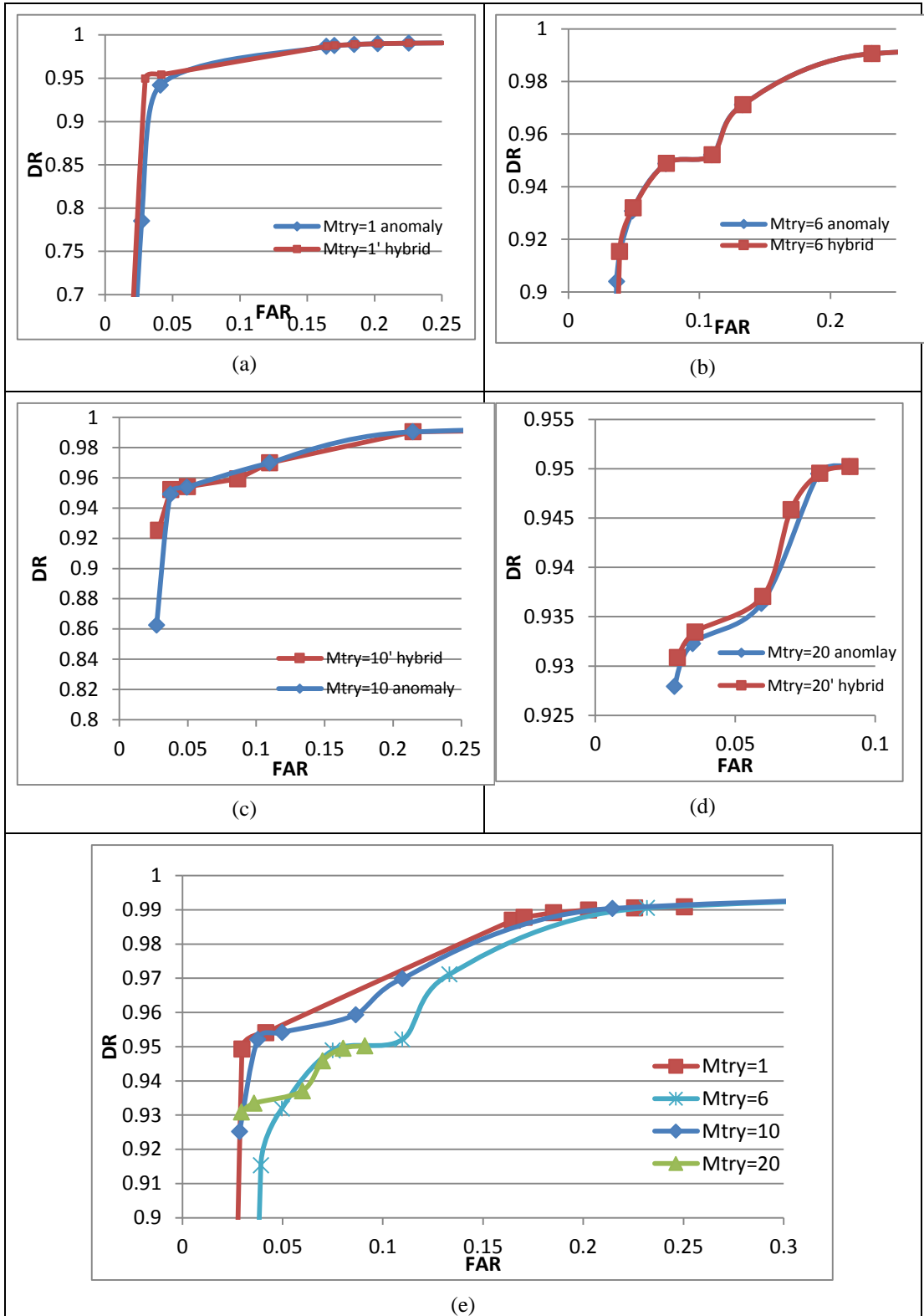


Figure 4.7 ROC Curve for HNB-OCC applied EMD-10% KDDCUP Dataset. (a) ROC curves for hybrid and anomaly models when parameter Mtry= 1. (b) ROC curves for hybrid and anomaly models when parameter Mtry= 6. (c) ROC curves for hybrid and anomaly models when parameter Mtry= 10. (d) ROC curves for hybrid and anomaly models when parameter Mtry= 20. (e) ROC curves for hybrid model with different values of Mtry

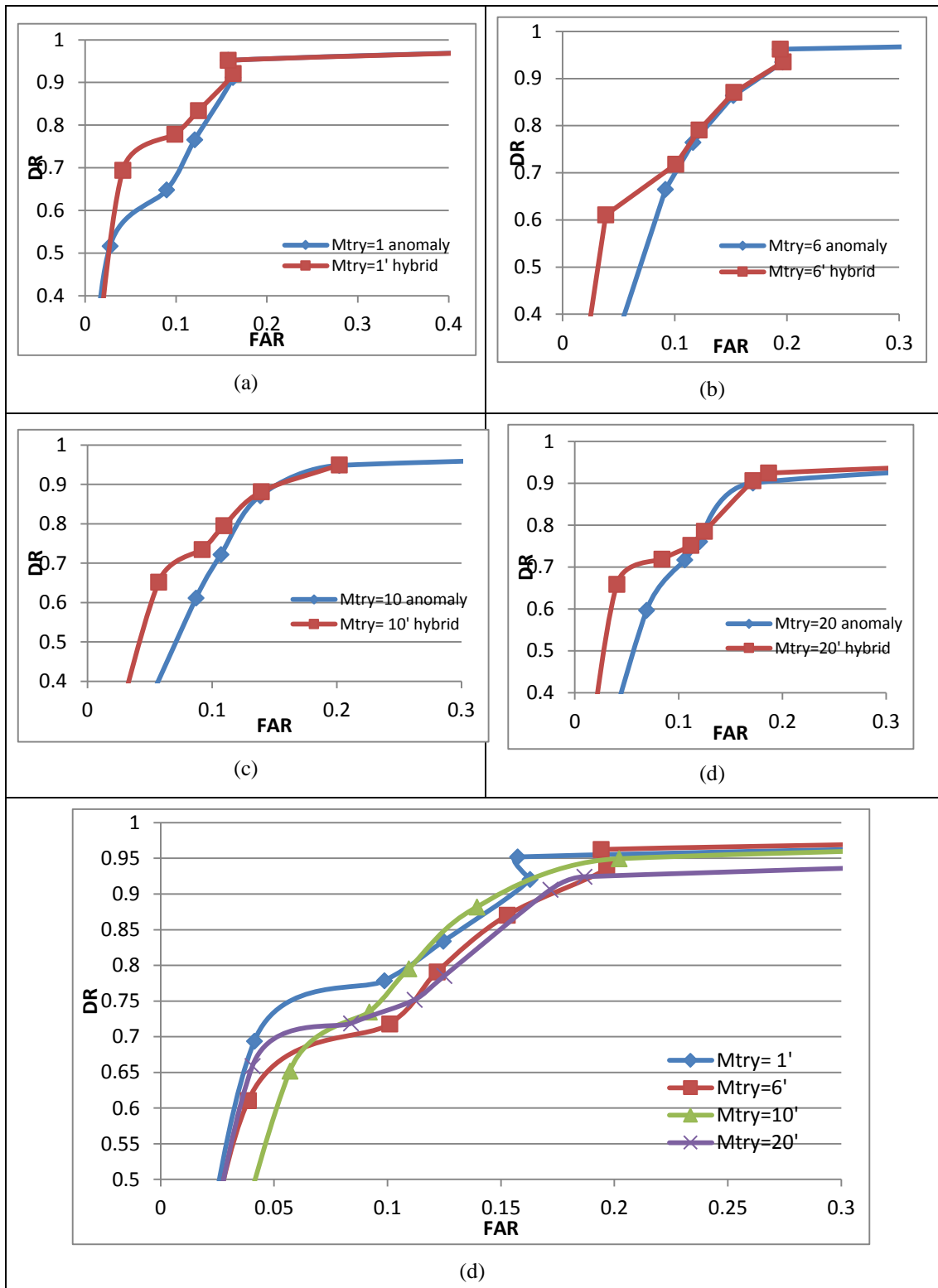


Figure 4.8 ROC Curve for HNB-OCC applied to EMD-NSL-KDD Dataset (a) ROC curves for hybrid and anomaly models when parameter $Mtry= 1$. (b) ROC curves for hybrid and anomaly models when parameter $Mtry= 6$. (c) ROC curves for hybrid and anomaly models when parameter $Mtry= 10$. (d) ROC curves for hybrid and anomaly models when parameter $Mtry= 20$. (e) ROC curves for hybrid model with different values of $Mtry$

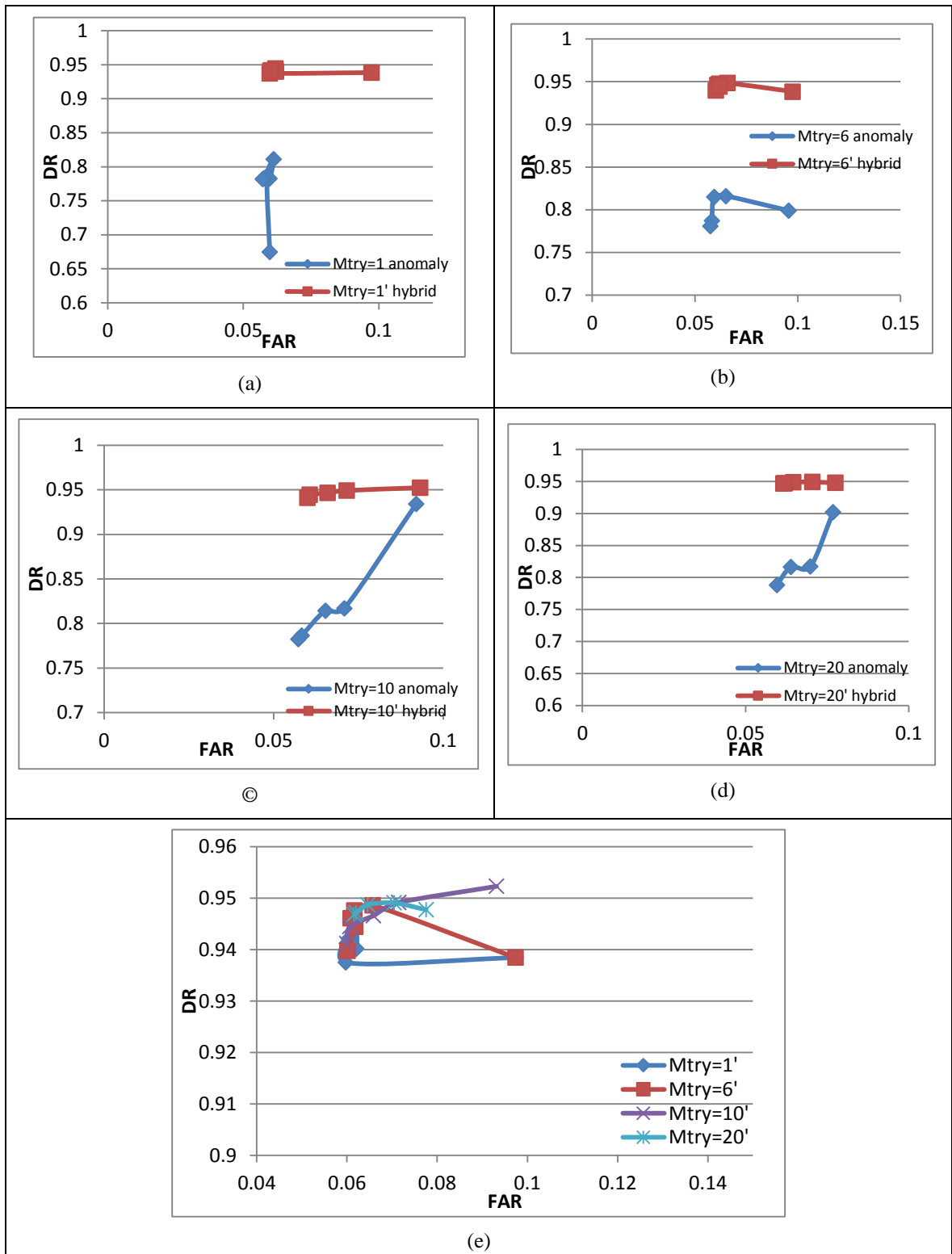
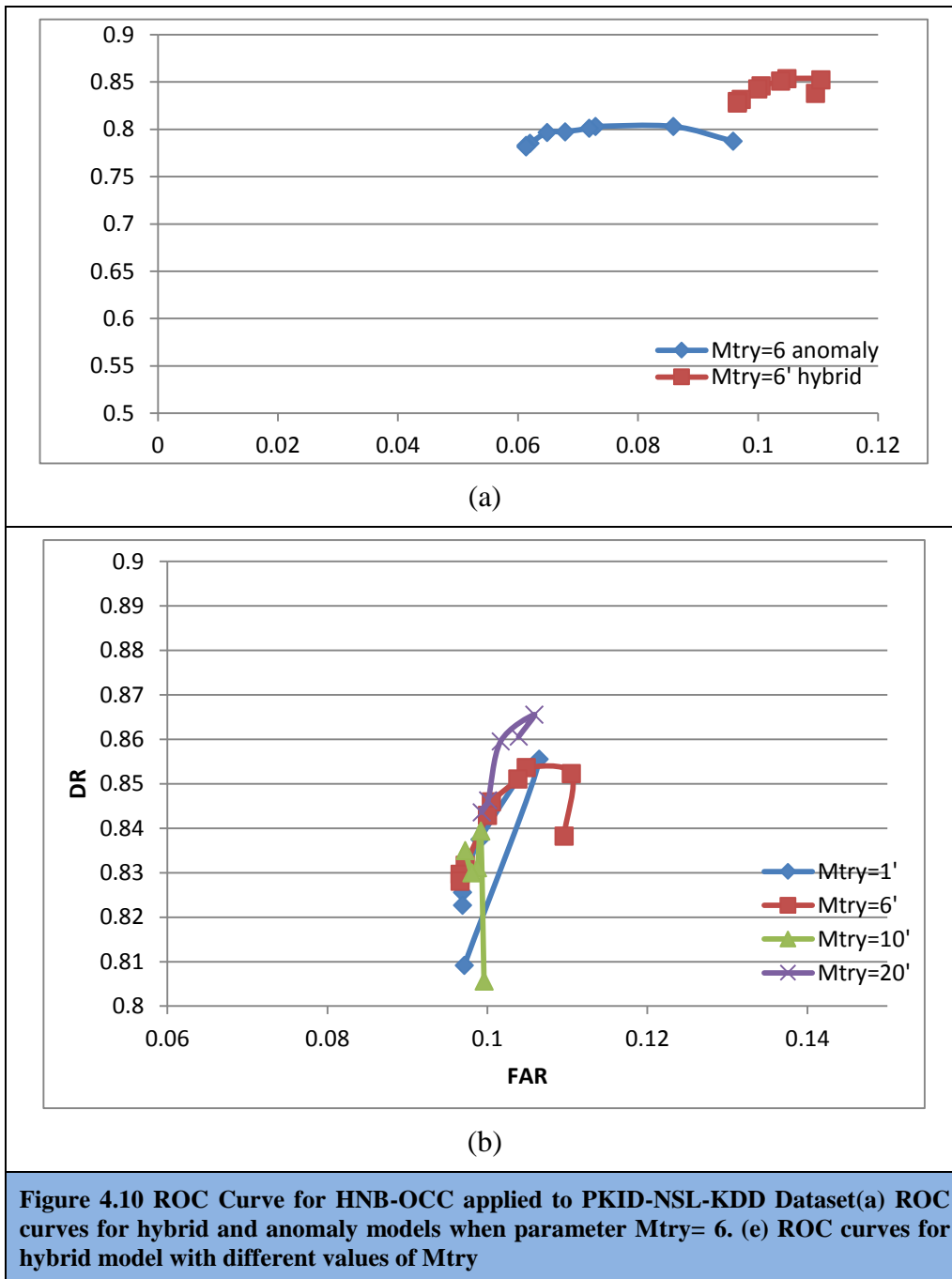


Figure 4.9 ROC Curve for HNB-OCC applied PKID-10% KDDCUP Dataset (a) ROC curves for hybrid and anomaly models when parameter $M_{try}=1$. (b) ROC curves for hybrid and anomaly models when parameter $M_{try}=6$. (c) ROC curves for hybrid and anomaly models when parameter $M_{try}=10$. (d) ROC curves for hybrid and anomaly models when parameter $M_{try}=20$. (e) ROC curves for hybrid model with different values of M_{try}



Different values of (*n_{tree}*) are tested for our model, earlier studies discuss this issue; [4] uses different values (10, 15, 20,25, 30, 35, 40, 45, and 50) to select the best models, Oshiro et al [63] suggest a range between 64 and 128 trees in a forest to obtain a good balance between performance, processing time, and memory usage. And different values is optimized according to application.

For the proposed hybrid model for **EMD data** initially starting from small *ntree* value, increasing *ntree* parameter value will increase the performance (DR) and at the same time FAR increases with acceptable range. Growing forest to medium values has no significant performance gain but only increases computational cost (time & memory) and error rate; continuous growing a larger forest up to several hundreds of trees it will usually overtraining -which vanishes in case of an infinite number of trees as stated in [58]. The parameter values where the model has no noticeable improvement and overfit is depends on dataset itself and *Mtry* value. Table 4.7 and 4.8 are an examples of EMD data. Table 4.7 shows DR and FAR for different *ntree* values when *Mtry* = 1 for 10% KDDCup dataset and Table 4.8 shows DR and FAR for different *ntree* values when *Mtry* = 10 for NSL-KDDCup. Tables 4.7 and 4.8 shows progressive DR increasing as *ntree* increasing; the increase amount of DR caused by moving between higher values of *ntree* is smaller with respect to the addition value to FAR, for example DR increased by (0.0003) as the value of *ntree* changes from 100 to 200 with 0.03 addition to FAR for 10%KDDCup dataset; overfitting occurs at 300 trees for 10%KDDCup dataset and at 200 for NSL-KDD dataset.

Table 4.7 DR and FAR for EMD 10% KDDCup when varied *ntree* and *Mtry*=1

<i>ntree</i>	DR	FAR
5	0.949	0.030
10	0.954	0.042
15	0.987	0.164
20	0.988	0.170
30	0.989	0.185
50	0.990	0.203
100	0.991	0.226
200	0.991	0.250
300	0.989	0.204
400	0.974	0.147

Table 4.8 DR and FAR for EMD NSL-KDD when varied *ntree* and *Mtry*=10

<i>ntree</i>	DR	FAR
15	0.735	0.092
10	0.795	0.109
20	0.882	0.139
100	0.950	0.202
200	0.952	0.177
300	0.950	0.161
400	0.953	0.152
500	0.950	0.152

For PKID data; the effect of changing *ntree* parameter value is different form EMD data; where FAR start higher when *ntree* parameter value is small and decrease as *ntree* increases. We reach stable point with smaller number of trees compared to EMD. In our case, PKID reach stable value and does not give flexibility that found when dealing with EMD data. Table 4.9 shows sample of DR and FAR for different *ntree* when *Mtry* = 1 for PKID discretization with 10%KDDCup dataset. As shown in Table 4.9 that DR values reaches 0.94 and still vibrate around without forwarding where in EMD different values can be reached by changing *ntree* parameter value.

Table 4.9 DR and FAR for PKID 10%KDDCup when varied *ntree* and *Mtry*=1

<i>ntree</i>	DR	FAR
5	0.938	0.097
10	0.938	0.060
20	0.944	0.062
30	0.940	0.062
50	0.942	0.060
100	0.941	0.060

Table 4.10 shows time estimation for different values of *ntree* and *Mtry* for EMD NSL-KDD dataset. As we show in the Table 4.10 time is (approximately) linearly increasing as we increase *ntree* value. Time estimation of the hybrid models equals the sum of

misuse and anomaly detection phase for training and testing steps; the time estimation for misuse phase is small and fixed with changing parameters for anomaly model and so it can be neglected.

Increasing $Mtry$ value will delay reaching convergence value. As Table 4.10 show that as we increase $Mtry$ often the values of DR will decrease with the same $ntree$ value, an example is the DR obtained when using $Mtry=1$ and $Mtry= 10$ with $ntree =100$; therefore to reach the value $DR=X$ we can reach with specific value ($Mtry, ntree$) you need to increase $ntree$ as the value of $Mtry$ increase to reach the same value X.

Table 4.10 Time estimation for hybrid models applied to NSL-KDD – EMD dataset

$(Mtry, ntree)$	Estimated Time	DR	FAR
0,10	35.32	0.791	0.122
0,20	65.41	0.870	0.153
0,100	270.71	0.962	0.194
0,200	533.56	0.951	0.155
1,100	209.39	0.952	0.157
10,100	327.17	0.950	0.202
20,100	488.27	0.906	0.172

4.6.3 Hybrid Framework with one-class SVM

This section shows and discusses the results for hybrid model used HNB with one-class SVM as outlier detection algorithm.

At training step; after building HNB model, normal instances are used to build patterns for anomaly detection using one-class SVM. HNB model used to evaluate a test set, then instances that classified as normal are fed into the one-class SVM anomaly detection model to determine outlier according to built SVM hyper-plane. To compare our proposed hybrid framework HNB-SVM, we use one-class SVM algorithm as anomaly detection, again normal class instances used to learn the system and built patterns and test set is evaluated over the built patterns.

In our study we optimize parameters for ν and g for RBF Gaussian kernel related to one-class SVM. We build patterns of normal data using different values for kernel parameter gamma (g) varied from 0.0001 to 1 (1,0.1, 0.024(*default value*),0.003 (recommended by [7]),0.001,0.0001) and with different values for parameter ν (0.5,0.1,0.05,0.01,0.001) and detect/determines outliers (novel attacks often) from the filtered test set using the built patterns. We start with NSL-KDD dataset for the first time to study the effect of changing g parameter value; the ROC curves for different values for g and ν are shown in Figure 4.11 for EMD data and Table 13 shows the results of anomaly one-class SVM and Hybrid IDS models using HNB-SVM, for $\nu=0.01$ and different value of RBF Kernel parameter g . As shown in Figure 4.11 and Table 4.11; when g is high and equal to 1 DR gives approximately full detection rate equal to 0.99, but with above 0.6 FAR which is undesirable and rejected. We notice that values (0.0001, 0.003, 0.001, 0.024) result convergent values of (DR, FAR) and this shown in Figure 4.11. As value of g go smaller no benefit is obtained and only small decrease on DR compared to default value of g is achieved as shown in Table 4.11 for values $g=0.003,0.001$ and 0.0001. So from this experiment and for next experiments used one-class SVM we only test g values (0.1 and 0.024) because these values gives different results, we choose 0.024 from other values (0.0001,0.003,0.001) because it is the default values $g=1/n$ where n is $=1/41=0.024$ and reported in many studies in different fields.

Table 4.11 shows the effect of changing value of g on both anomaly detection using one-class SVM and the proposed hybrid HNB-SVM, the effect is homogeneous for both models but with the difference caused by misuse phase.

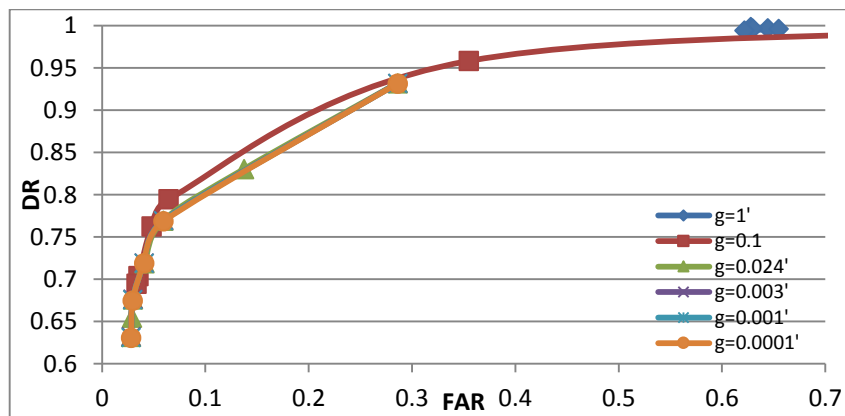


Figure 4.11 Comparison between results of EMD NSL-KDD model using different values of RBF Kernel g

Table 4.11 Results of anomaly one-class SVM and Hybrid IDS using HNB-SVM, for different value of RBF Kernel parameter g and $\nu = 0.01$

	one-class SVM 0.01		HND-SVM 0.01	
	DR	FAR	DR	FAR
$g= 1$	0.996	0.655	0.996	0.655
$g=0.5$	0.991	0.315	0.991	0.315
$g=0.1$	0.622	0.026	0.704	0.035
$g=0.024$	0.568	0.013	0.676	0.030
$g=0.003$	0.568	0.013	0.677	0.030
$g=0.001$	0.567	0.013	0.676	0.030
$g=0.0001$	0.565	0.013	0.675	0.030

As mentioned above, the decision boundaries of one-class SVM models using a Gaussian kernel are controlled by the parameter g . When g is small (0.024) it profiles normal data too widely (repeated and rare), so small number of instances will classified as outlier and higher number of anomaly classified as normal which lead to small DR and error rate; as g increases (0.1) decision boundary become more flexible, DR will becomes higher this because profiles data less wider/much narrower so that number of instances correctly classified as outlier form attack instances will be higher and also boundary will consider more normal instances as outlier so FAR is faced small increase. For higher values of g (0.5,1), DR is approximately equals to one because its appears to profiles normal data too narrowly, only frequent points are profiles so higher number of normal data be classified as outlier and all attack instances will be classified as outliers. Figure 4.11 and Table 4.11 show an example of the previous discussion.

Optimal values for different models used hybrid model with one–class SVM are shown in Table 4.12; medium values of g (0.1) achieves the optimal values for DR and FAR for both EMD and PKID discretized datasets.

Table 4.12 Optimum parameters for one-class SVM hybrid

10%KDDCup-EMD	$g= 0.1$	Figure 4.12 (diamond mark curve)
10%KDDCup-PKID	$g= 0.1$	Figure 4.13 (rectangular mark curve)
NSL-KDD-EMD	$g= 0.1$	Figure 4.14 (rectangular mark curve)
NSL-KDD-PKID	$g= 0.1$	Figure 4.15 (rectangular mark curve)

Figures 4.10 shows the ROC Curves for HNB-SVM applied to 10% KDDCup dataset discretized using EMD method; the figure plots ROC curves of hybrid model HNB-SVM (shown in red color) and anomaly detection using one-class SVM (blue curve), with g values equal to 0.024 and 0.1; the last sub-figure compares between the ROC curves obtained by different values of g for the proposed hybrid model HNB-SVM. Figures 4.12-4.15 show the ROC curve of hybrid models 10%KDDCup_PKID, NSL-KDD_EMD and NSL-KDD_PKID respectively, with the same arrangement shown in Figure 4.10.

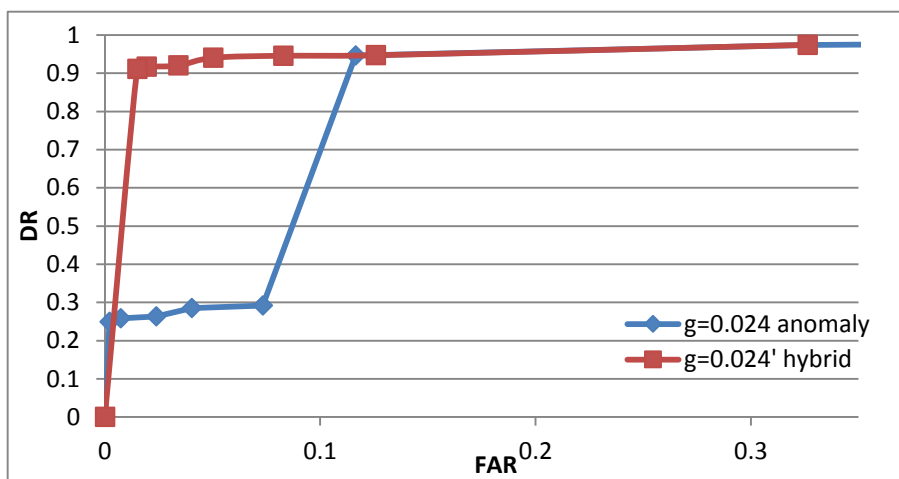
As Figures 4.12-4.15 show that using hybrid model HNB-SVM gives better results than using conventional one-class SVM as anomaly detection for IDS datasets NSL-KDD and 10%KDDCup.

- ✚ **For 10%KDDCup dataset, as shown from Figures 4.12 and 4.13**, for lower values of ν higher difference reaching more than 50% of detection rate values between proposed model and conv. one-class SVM, this is still until higher values of ν applied where convergent values is appears. With our proposed hybrid method for both PKID and EMD we get DR about 94% with about 5%, and reaching to 98% of DR needs more victims from normal connection reach 30% occurs when higher values of ν parameter equals to 0.5.
- ✚ **As shown in Figures 4.14 and 4.15 that shows the ROC curves for NSL-KDD dataset**, conv. one-class SVM is better when DR is lower than 70% which is occurred at small values of ν , as ν increases (at moderate values) DR and FAR of the proposed algorithm becomes better than conv. one-class SVM with difference starting high (reaches 5%) and this difference decrease and finally disappears for higher values of ν .

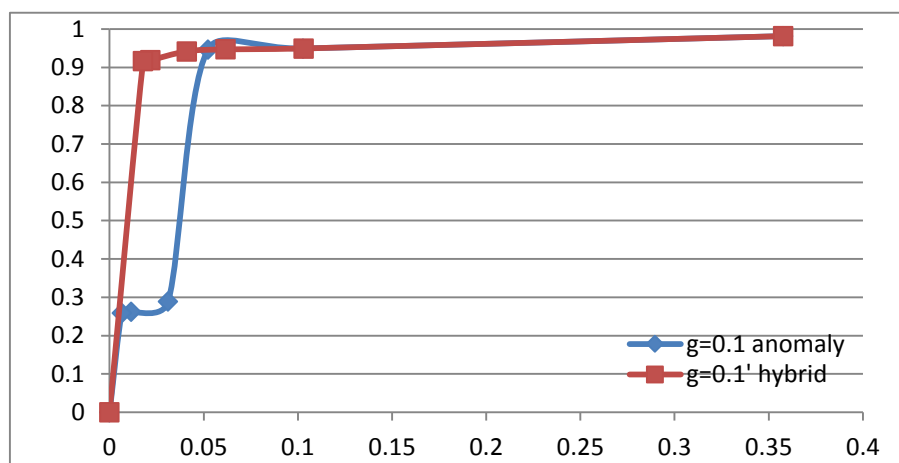
For smaller values of ν parameter, DR of the proposed hybrid model for **EMD data** is over 75% achieved with FAR does not exceed 5%, this means that DR is increased by $(0.75-0.57=0.18=18\%)$ with increment about 2% in FAR, where 0.57 is the DR of the misuse detection with 0.027 FAR. The increase of DR causes FAR to be higher, reaching 95% DR causes FAR to become about 35% as shown in Figure 4.14. The same behavior for **PKID data** illustrated by Figure 4.15, with difference that it starts with higher FAR compared to EMD for smaller values of ν . As shown in the figure we can reach 75% DR with 8% error.

This behavior is acceptable since one-class SVM model only focuses on the most frequent training patterns when ν is high, In contrast, if ν is very low, the decision hyper plane contains most training instances, including noisy data.

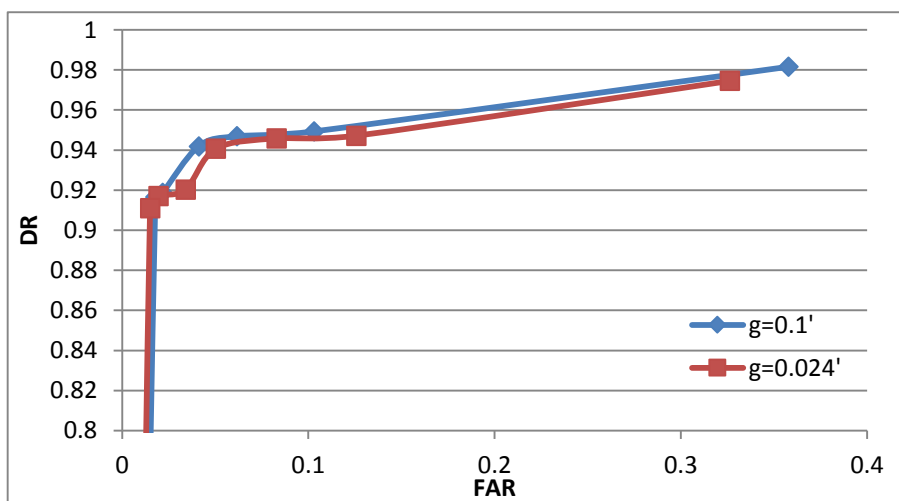
Finally if we review the difference between NSL-KDD and 10%KDDCup behavior when hybrid IDS (HNB-SVM) and anomaly IDS (one-class SVM) are used, we notice that the deference between HNB-SVM and one-class SVM for 10%KDDCup outperforms NSL-KDD, and the difference is shown obviously in Figures 4.12-4.15; the difference for 10%KDDCup can reach more than 50% in DR at the same FAR value, where this difference is less obvious in NSL-KDD. The reason for this is that misuse detection phase behavior difference. Where at NSL-KDD zero point for hybrid system DR is equal to HNB misuse = 0.57 with FAR equals 0.0276; and DR equals 0.905 with 0.014 FAR is the zero point for 10%KDDCup.



(a)

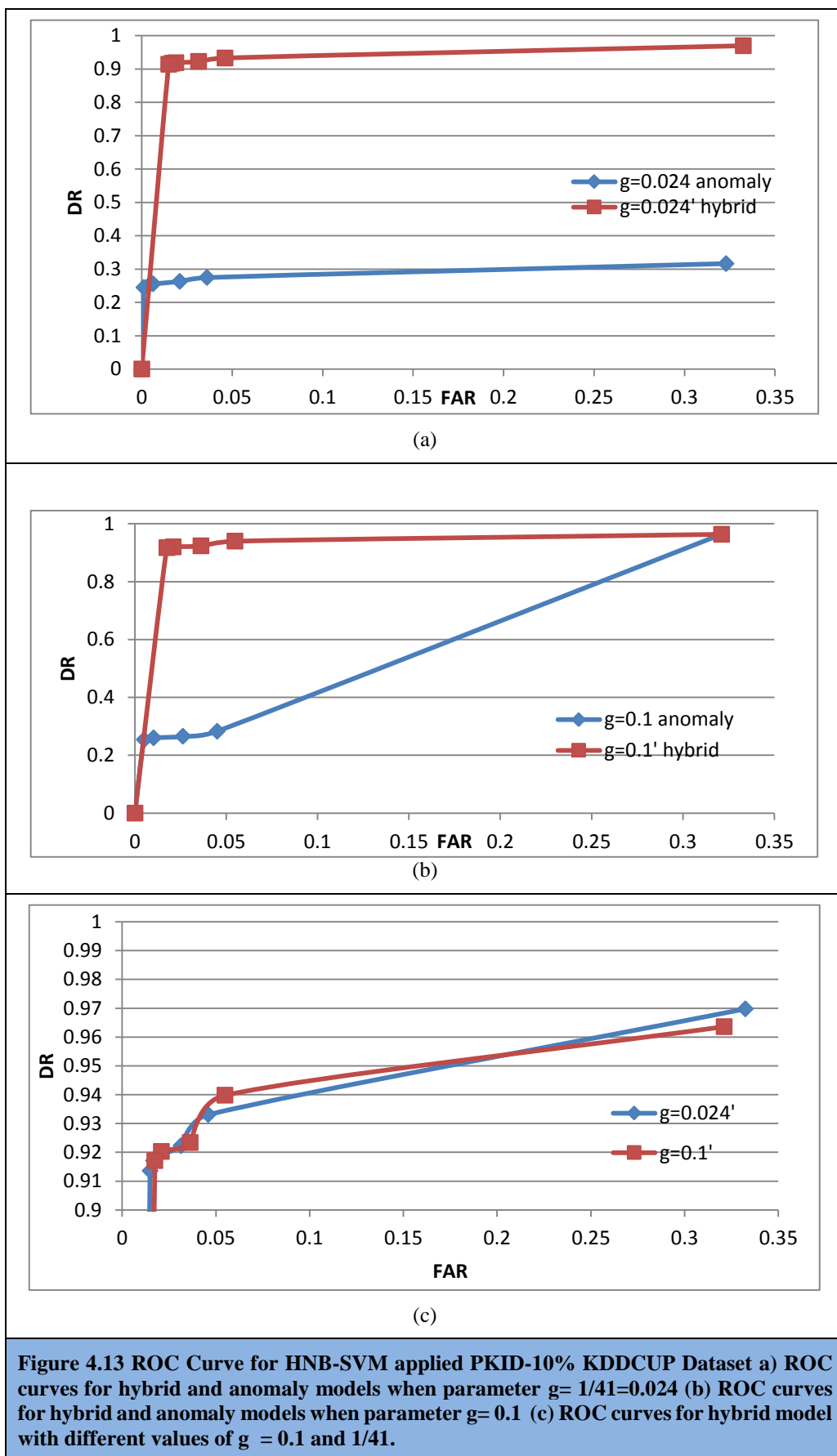


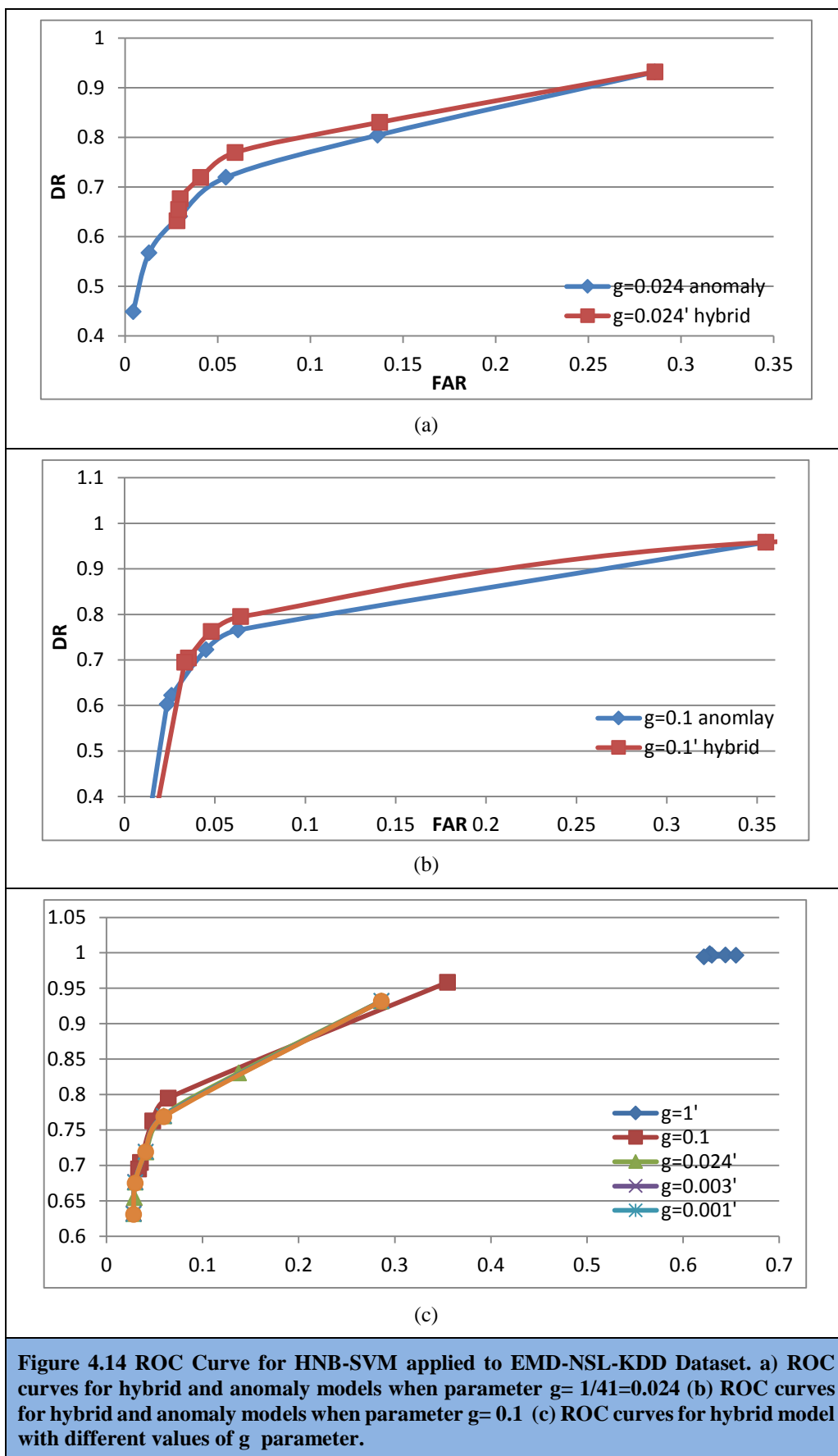
(b)

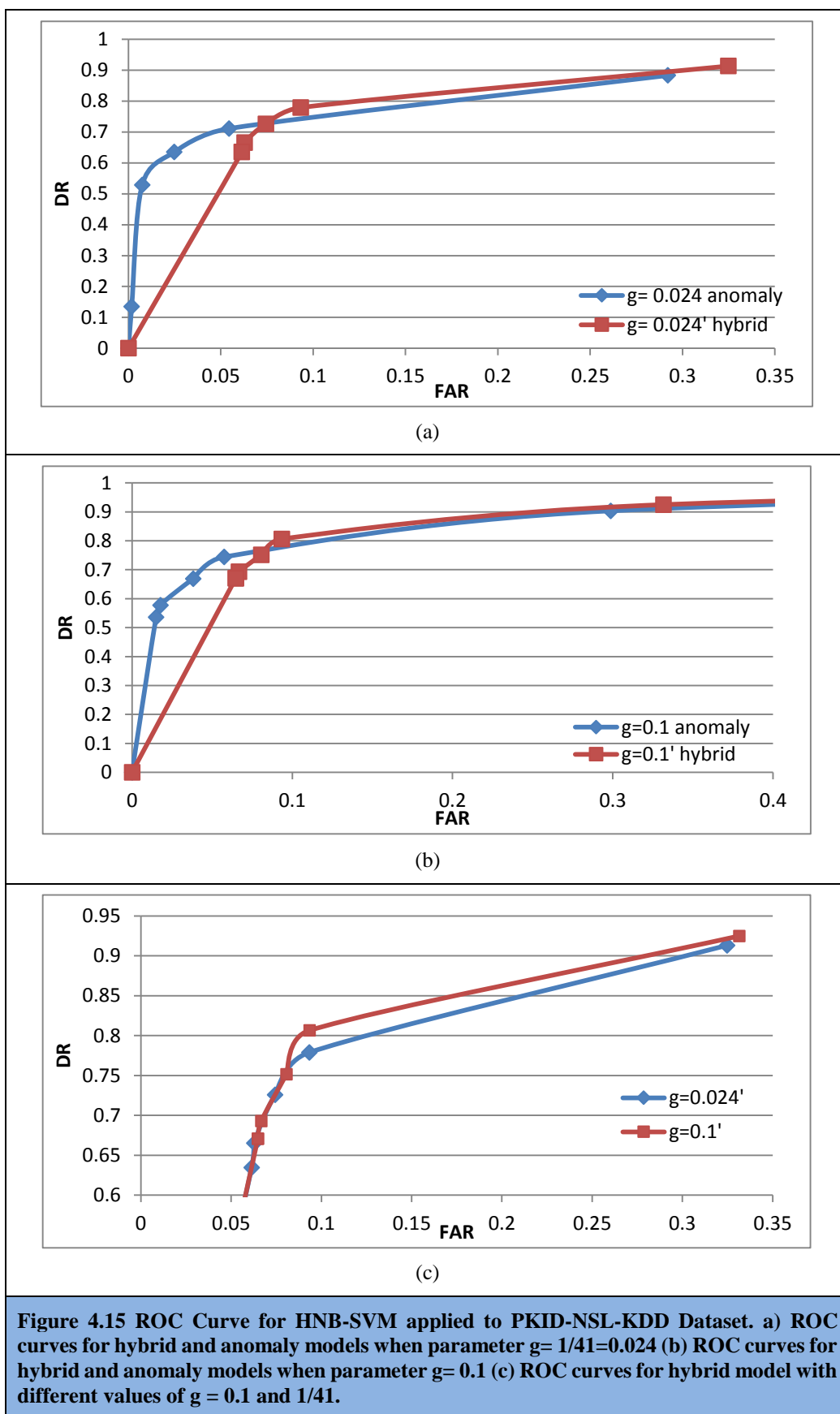


(c)

Figure 4.12 ROC Curve for HNB-SVM applied EMD-10% KDDCUP Dataset (a) ROC curves for hybrid and anomaly models when parameter $g = 1/41 = 0.024$ (b) ROC curves for hybrid and anomaly models when parameter $g = 0.1$ (c) ROC curves for hybrid model with different values of $g = 0.1$ and $1/41$.







4.6.4 Comparison between Hybrid models

Figures 4.16- 4.19 compares between HNB-OCC and HNB-SVM hybrid models applied to 10% KDDCUP EMD, 10%KDDCup PKID, NSL-KDD EMD, and NSL-KDD PKID datasets, respectively. Optimal curve for each model is used in this comparison.

Table 4.13 compares the time consumption using approximately equally DR selected points from ROC curves obtained by applying the hybrid models (HNB-OCC and HNB-SVM) to 10%KDDCup and to NSL-KDD datasets. From Table 4.13 and Figures 4.16-4.19 we can say that:

1. Data Discretized using EMD:

- ✚ For 10%KDDCup EMD dataset as shown in Figure 4.16, HNB-OCC model outperforms HNB-SVM model in terms of DR and FAR.
- ✚ For NSL-KDD EMD dataset as shown in Figure 4.18, HNB-SVM model is better than HNB-OCC for lower detection rates, but HNB-OCC model is return better for medium and higher detection rates.
- ✚ Time consumption for HNB-OCC is extremely small comparing to HNB-SVM model as shown in Table 4.13.

2. Data Discretized using PKID:

- ✚ Results of PKID discretized data with HNB-OCC model tends to be stable. Obtained results are usually closet and convergent in small region of DR and FAR as we mentioned in section 4.6.2. As we show in Figure 4.17 for **10%KDDCup dataset**, obtained values are between 0.94 and 0.95 DR with FAR between 0.06 and 0.08. These values are better than or convergent with HNB-SVM ROC curve (over the ROC curve of HNB-SVM). For **NSL-KDD dataset** as shown in Figure 4.19, the obtained values are between 0.80 and 0.85 DR with FAR between 0.09 and 0.1. These values are also better than or convergent with HNB-SVM ROC curve.
- ✚ HNB-OCC with PKID data fails to reach some higher values that can be reached using HNB-SVM. As we show in Figure 4.19, HNB-SVM reaches 0.925 DR with 0.33 FAR where HNB-OCC fails to reach convergent value.
- ✚ On the other side, HNB-OCC with PKID data fails in case of smaller values of DR; take as an example in Figure 4.17 HNB-SVM reaches value of 0.92 DR with 0.036 FAR which cannot be reached in our case when using Mtry=10.
- ✚ Time consumption to reach specific value of DR using HNB-OCC is small comparing to HNB-SVM model as shown in Table 4.13.

From the above discussion we can conclude to:

- ✚ Time consumption for HNB-OCC is better than HNB-SVM model.
- ✚ HNB-OCC model usually outperforms HNB-SVM model in terms of DR, FAR when acting with EMD data.
- ✚ HNB-SVM model is stable in dealing with PKID and EMD discretized data

- ✚ HNB-OCC model with EMD data is the leading model amongst all models, best DR and FAR can be reach with it.
- ✚ Rather than accepted values reached using HNB-OCC model using PKID data, it is failed to reach higher and lower detection rates.

Table 4.13 Time comparison between different proposed hybrid models

model	OCC-RF			One-class SVM		
	DR	FAR	Time (Second)	DR	FAR	Time (Second)
10%KDDCup EMD	0.987	0.165	98	0.981	0.358	3790
10%KDDCup PKID	0.944	0.062	86.5	0.940	0.055	1905
NSL-KDD EMD	0.952	0.157	198.5	0.958	0.355	1545.5
NSL-KDD PKID	0.809	0.097	18	0.806	0.093	897

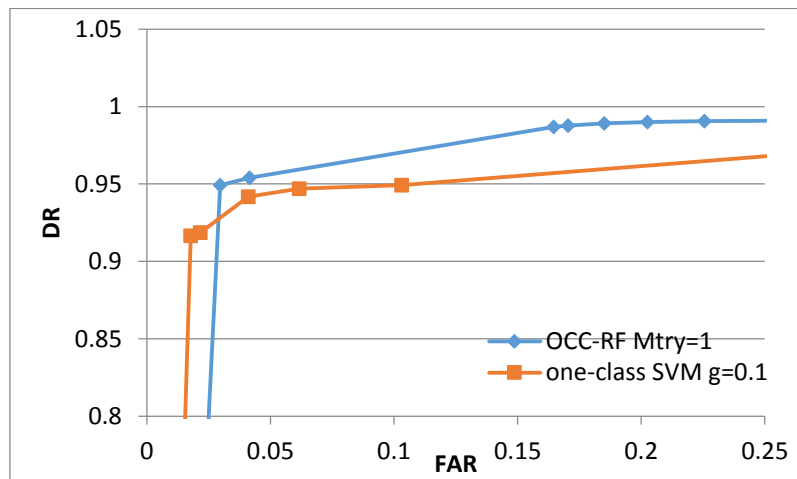


Figure 4.16 Difference between HNB-OCC and HNB-SVM applied 10% KDDCUP-EMD Dataset

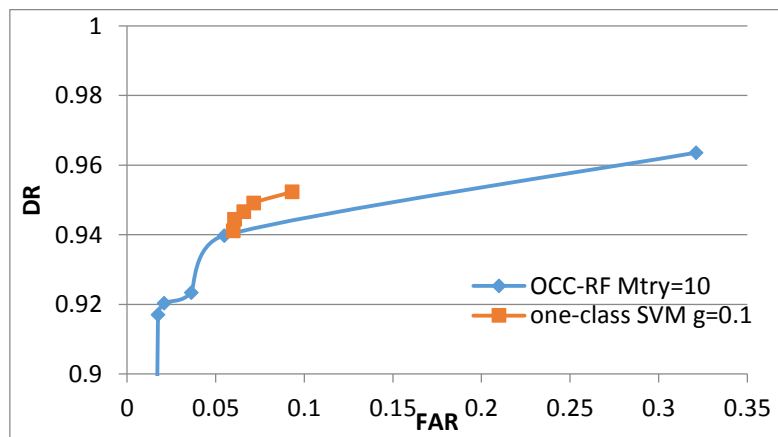


Figure 4.17 Difference between HNB-OCC and HNB-SVM applied 10% KDDCUP-PKID Dataset

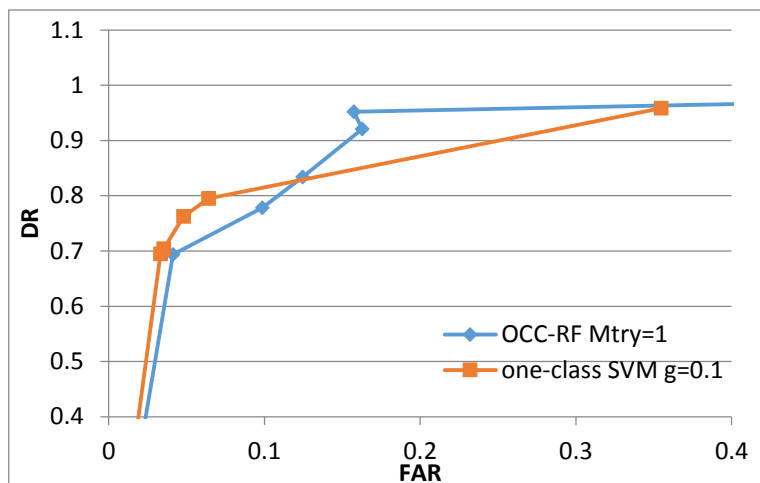


Figure 4.18 Difference between HNB-OCC and HNB-SVM applied NSL-KDD EMD Dataset

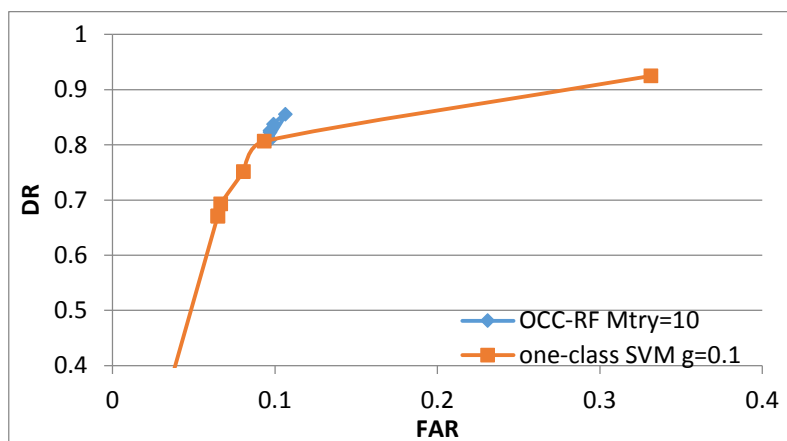


Figure 4.19 Difference between HNB-OCC and HNB-SVM applied NSL-KDD PKID Dataset

4.7 Improved Mixed Model using Clustering

This section shows and discusses the experiments performed using hybrid models with k-means clustering algorithm to improve the performance.

4.7.1 Parameter Optimization for Clustered Anomaly Detection

In our experiments different values for parameters of outlier detection algorithms are evaluated (i.e. OCC-RF and one-class SVM) on the level of clusters. We note from our experiments that there exists collection of parameters values that can contribute in reaching values over the ROC curve obtained by hybrid framework presented in the previous section, but not all combinations will lead to better values; this is obvious because we built k models and each model will build depending on different training data, where each model uses the normal instances related to a cluster. Parameters play the main role in building the models, for example for cluster i may $Mtry=1$ and $nree = 2$ using OCC-RF model gives the best value in anomalies detection and the same parameter may fail for cluster j and so on. Parameter sensitivity is due to the nature of instances related to that cluster and the trivial behavior of used algorithm (OCC-RF or one-class SVM in our study). So choosing best combination of parameters – for each cluster- is the main step here.

Parameters of clustering model for OCC-FR ($Mtry$, $nree$) and one-class SVM (ν , g) is choose manually, so that each cluster is tested individually using range of parameters for best output using test set, only best parameters will take in the calculations.

4.7.2 Discussion

The range of parameters used is control the output of the model, best choices may be appears when the range used for parameter is wider. For our experiment the values used for g in RBF kernel in one-class SVM and $Mtry$ in OCC-RF are choose to be distant from each other to give different behavior, as shown in Table 4.14. More test needed to optimize the best parameters. Values of parameter $nree$ are chosen as in the previous section and also the value of ν parameter in one-class SVM; where the values shown in Table 4.14 for different combinations.

Figures 4.20- 4.27 show the ROC curves of clustering model for different combinations of (datasets, discretization method, and outlier detection algorithm used), KDDCup_EMD_OCC, KDDCup_EMD_SVM, KDDCup_PKID_OCC, KDDCup_PKID_SVM, NSL-KDD_EMD_OCC, NSL-KDD_EMD_SVM, NSL-KDD_PKID_OCC, NSL-KDD_PKID_SVM, respectively. The results of the clustering models is shown as scattered points above the ROC curves obtained by the proposed hybrid model.

Figures show that clustering model can produce points on ROC curve (DR,FAR) over the curves of corresponding hybrid model, the clustering values better than in DR or FAR or in both.

Table 4.14 parameter range used for improved hybrid IDS

KDDCup_EMD_OCC	$Mtry=6,10$	$ntree= 2,5,10,20,50,100$
KDDCup_EMD_SVM	$g= 0 ; 0.0001$	$v=0.001,0.01,0.05,0.1,0.5$
KDDCup_PKID_OCC	$Mtry=1,6$	$ntree= 2,5,10,20,50,100$
KDDCup_PKID_SVM	$g= 0 ; 0.1$	$v=0.001,0.01,0.05,0.1,0.5$
NSL-KDD_EMD_OCC	$Mtry=1,6$	$ntree= 2,5,10,20,50,100$
NSL-KDD_EMD_SVM	$g= 0.1 ; 0.001$	$v=0.001,0.01,0.05,0.1,0.5$
NSL-KDD_PKID_OCC	$Mtry=6,20$	$ntree= 2,5,10,20,50,100$
NSL-KDD_PKID_SVM	$g= 0.1 ; 0$	$v=0.001,0.01,0.05,0.1,0.5$

4.7.3 Example of clusters

We will discuss two examples of clustering model. Table 4.15 and 4.16 is an examples of parameters selection for clustering model for NSL-KDD_EMD_OCC and KDDCup_PKID_SVM respectively. Table 4.15 shows parameter evaluation for improved hybrid model (HNB-OCC) using k-means clustering with $k = 10$, $Mtry = (0,1)$ and $ntree = (2,5,10,20,50,100)$; and Table 4.16 shows parameter combinations of the improved hybrid model (HNB-SVM) using k-means clustering with $k = 10$, $g = (0,0.1)$ and $v = (0.001,0.01,0.05,0.1,0.5)$. The first two columns in Table 4.15 and 4.16 shows the normal and attack instances for each one of the ten clusters for the test set - training set does not contain attack instances-, and other columns shows combinations of optimized parameters related to each cluster, the resultant DR and FAR for each combination is shown in the bottom of the column.

From tables we have some notations:

- ✚ We can get results using OCC-RF with fixing $Mtry$ while changing $ntree$ value as in Table 4.15 (R1,R2 obtained when $Mtry$ equals 6 and R3,R4 obtained when $Mtry$ equals 1) or by combine from different $Mtry$'s (R5 and R6).
- ✚ We can get results using one-class SVM with fixing RBF Kernel parameter g while changing v value as in Table 4.16 (R1-R4 obtained for default value of g and R5,R6 obtained when g equals 0.1) or by combine from different levels of RBF kernel.
- ✚ We can divide clusters to three types:
 1. Type 1: cluster approximately pure normal cluster or pure attack cluster (ex: cluster 1 in Table 4.15 and cluster 0 in Table 4.16); usually higher or lower values of parameter $ntree$ in OCC-RF or v in one-class SVM can get the best results. Cluster1 in Table 4.15 with ($ntree=2,5,10,20,50$) gives accepted results but with higher FAR or lower DR, best results is occurred at $ntree= 100$ the highest value tries here.

2. Type 2: some clusters has no weight with respect to others (ex: cluster7 in Table 4.16); choosing parameters for this type is time wasting, because it does not affect the total result so it can be neglected.
3. Type 3: mixed clusters; usually most of mixed clusters is easily determine the best parameters based on the difference between the correct and incorrect classified instances, but some is difficult to determine parameters because changing parameters is highly affect both DR and FAR, the reason for this is the higher degree of similarity between attack and normal instances in these clusters; cluster 7 in Table 4.15 is an example in R3 and R4 changing parameter cause higher changes in FAR, cluster 3 in Table 4.16 shows another example where at R5 and R6 changing v from 0.001 to 0.5 will cause higher change in DR. For these examples both parameters are accepted and results points outperform the proposed hybrid model, the same is applied to other clusters.

Table 4.15 Example of improved hybrid system (HNB-OCC) using k-means clustering with $k = 10$, $Mtry = (0,1)$ and $ntree = (2,5,10,20,50,100)$ applied to NSL-KDD EMD

	Normal	Attack	R1	R2	R3	R4	R5	R6
0	596	1663	0,50	0,50	1,20	1,20	0,50	0,50
1	2666	145	0,100	0,100	1,100	1,100	1,100	1,100
2	2556	1548	0,100	0,100	1,100	1,100	1,100	1,100
3	511	1090	0,100	0,100	1,20	1,20	1,20	1,20
4	42	259	0,2	0,2	1,2	1,2	1,2	1,2
5	255	66	0,10	0,10	1,5	1,5	1,5	1,5
6	22	105	0,20	0,20	1,50	1,50	0,20	0,20
7	749	147	0,10	0,2	1,1	1,2	1,2	1,1
8	1463	17	0,100	0,100	1,100	1,100	1,100	1,100
9	583	431	0,100	0,100	1,50	1,50	1,50	1,50
		DR	0.906	0.901	0.897	0.900	0.899	0.896
		FAR	0.152	0.149	0.101	0.138	0.137	0.099

Table 4.16 Example of improved hybrid system (HNB-SVM) using k-means clustering with $k = 10$, $g = (0,0.1)$ and $v = (0.001,0.01,0.05,0.1,0.5)$ applied to KDDCup PKID

	Normal	Attack	R1	R2	R3	R4	R5	R6
			g=0	g=0	g=0	g=0	g=0.1	g=0.1
0	2111	134	0.001	0.001	0.001	0.001	0.001	0.001
1	2200	8005	0.1	0.1	0.5	0.5	0.05	0.5
2	4102	2042	0.001	0.001	0.001	0.001	0.001	0.001
3	4723	3850	0.001	0.001	0.001	0.001	0.001	0.5
4	51	644	0.01	0.01	0.01	0.01	0.01	0.01
5	22274	177	0.01	0.01	0.01	0.01	0.01	0.01
6	7074	4715	0.001	0.5	0.001	0.5	0.05	0.5
7	1	1	0.001	0.001	0.001	0.001	0.001	0.001
8	0	51	0.001	0.001	0.001	0.001	0.001	0.001
9	17181	3075	0.05	0.05	0.05	0.05	0.01	0.01
DR			0.942	0.953	0.956	0.967	0.942	0.985
FAR			0.032	0.079	0.048	0.095	0.033	0.219

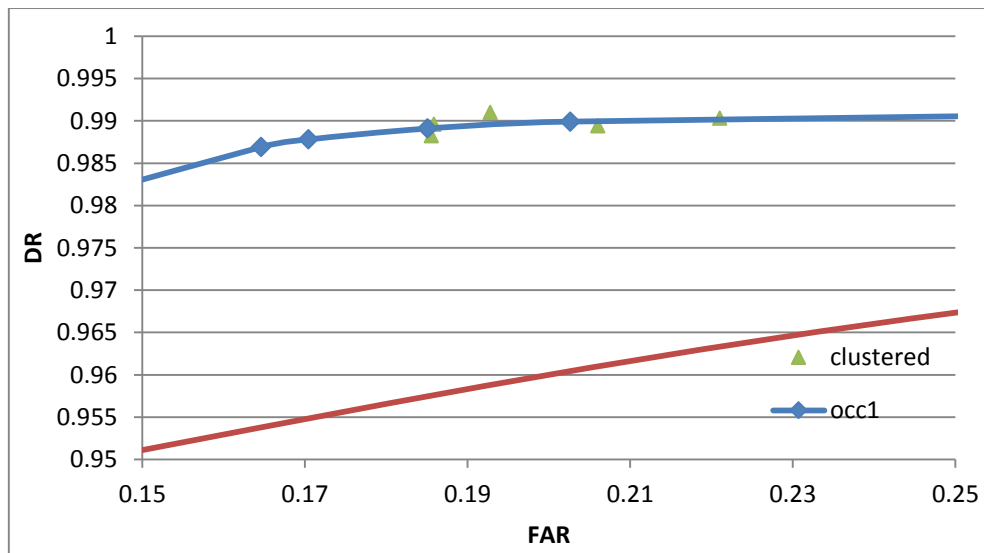


Figure 4.20 Results of improved hybrid IDS for KDDCup EMD using HNB-OCC model

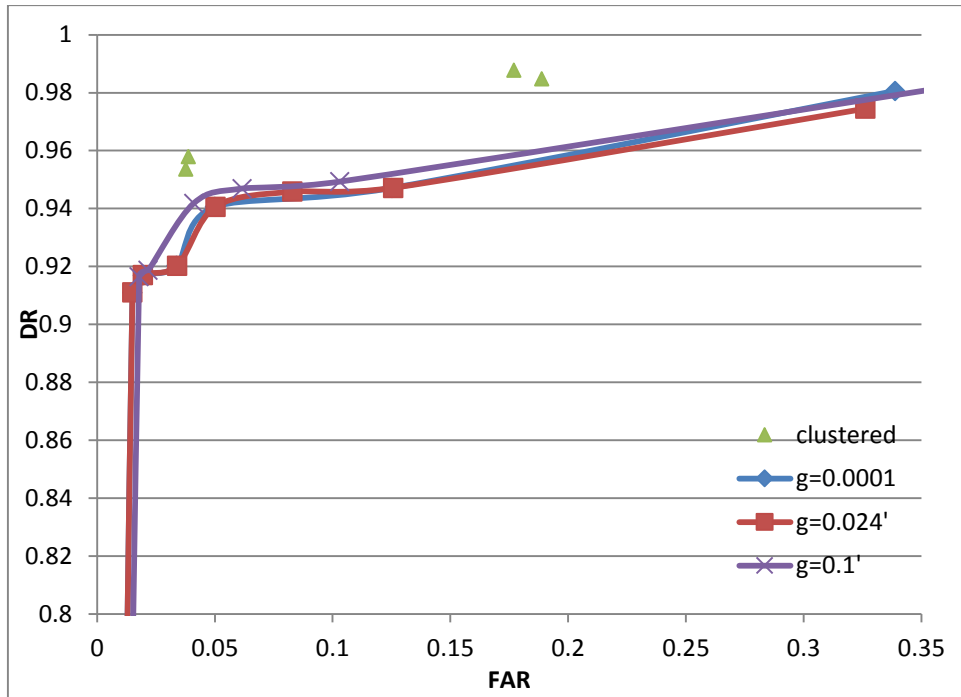


Figure 4.21 Results of improved hybrid IDS for KDDCup EMD using HNB-SVM model

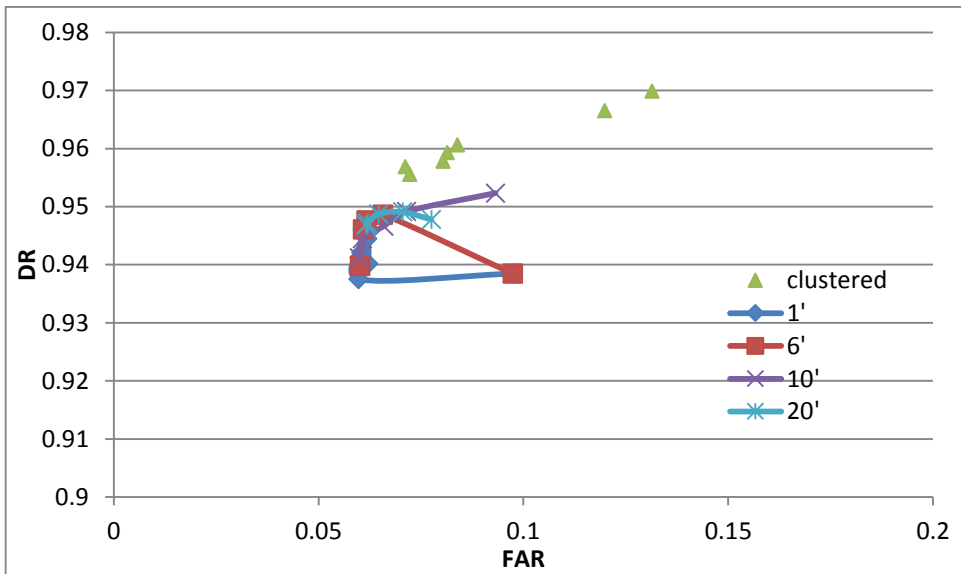


Figure 4.22 Results of improved hybrid IDS for KDDCup PKID using HNB-OCC model

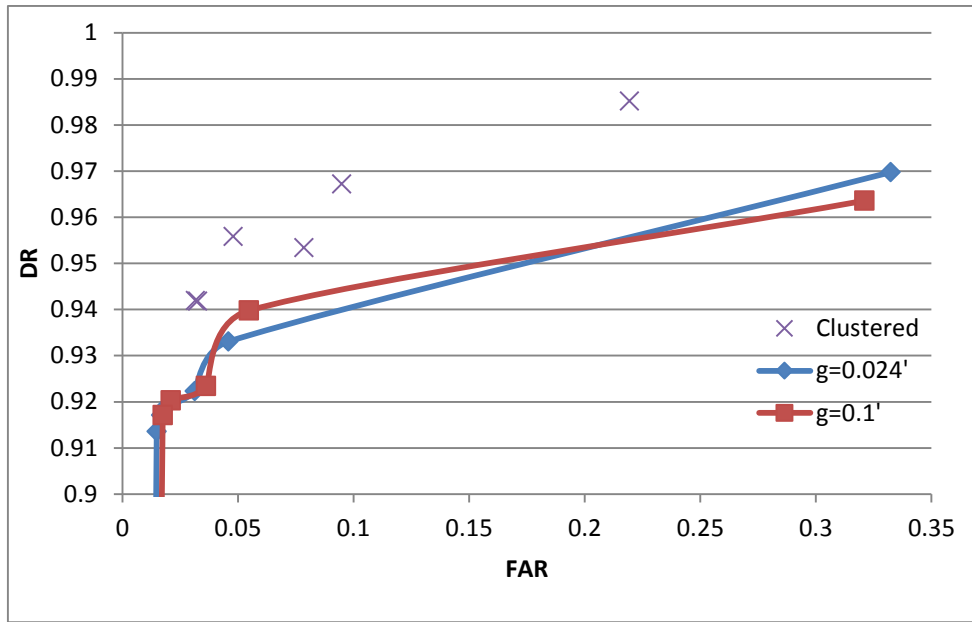


Figure 4.23 Results of improved hybrid IDS for KDDCup PKID using HNB-SVM model

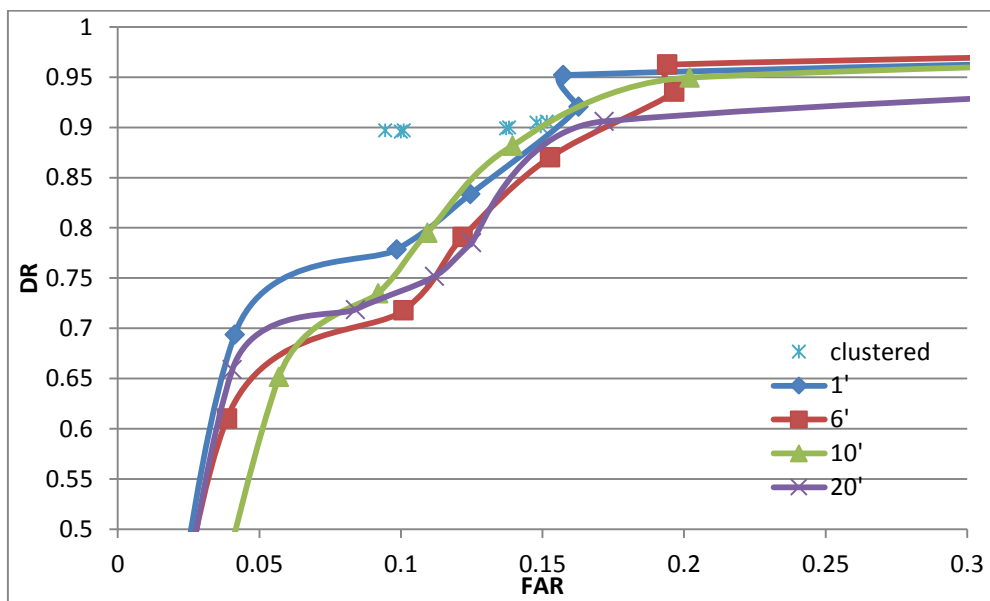


Figure 4.24 Results of improved hybrid IDS for NSL-KDD EMD using HNB-OCC model

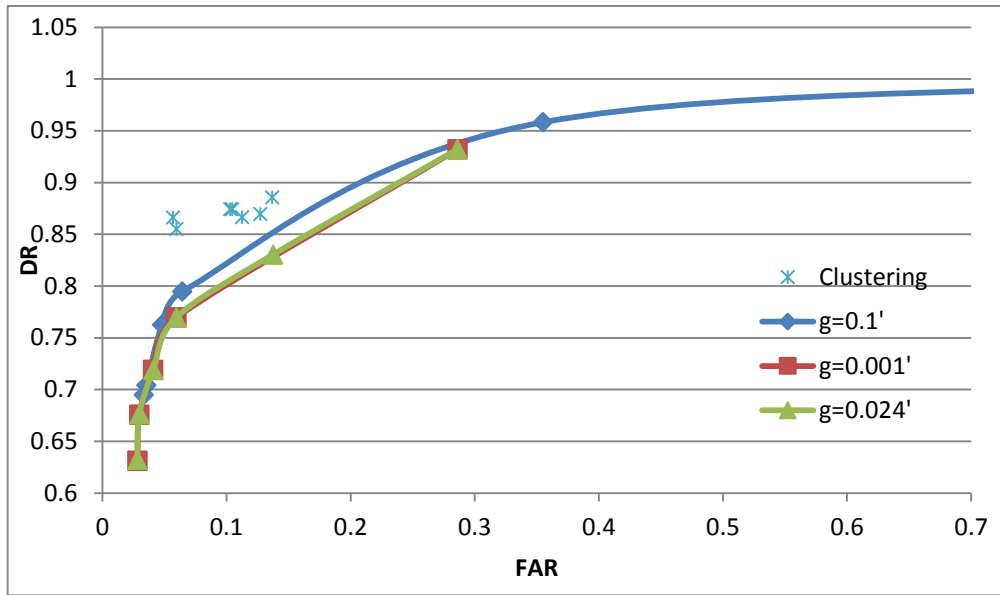


Figure 4.25 Results of improved hybrid IDS for NSL-KDD EMD using HNB-SVM model

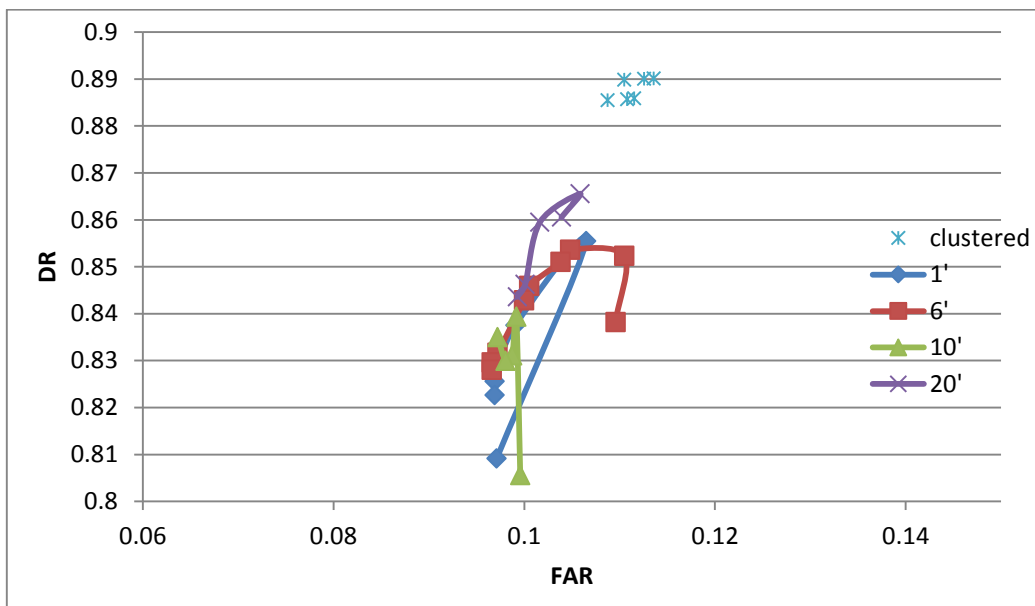


Figure 4.26 Results of improved hybrid IDS for NSL-KDD PKID using HNB-OCC model

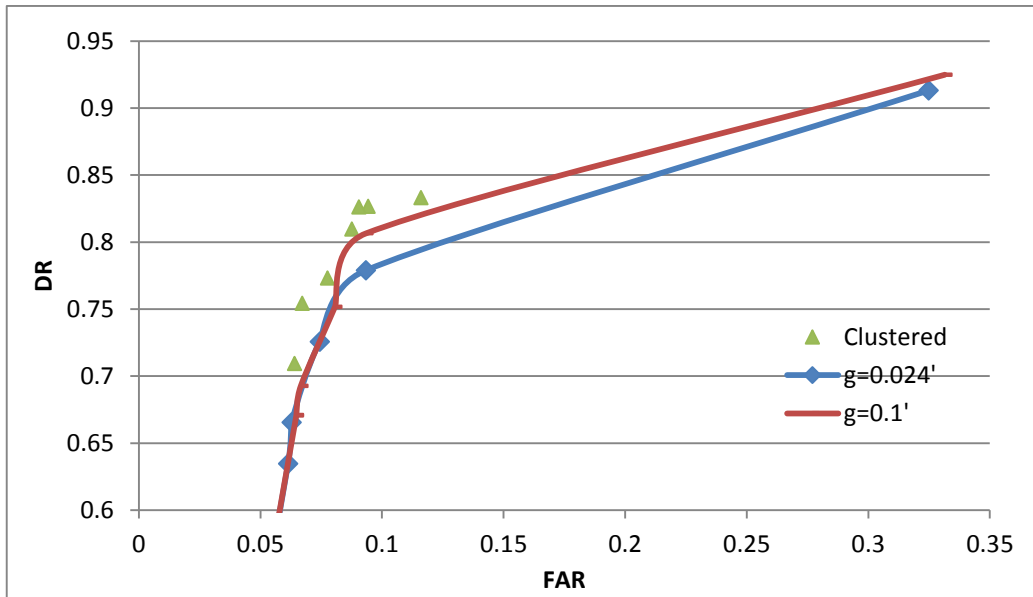


Figure 4.27 Results of improved hybrid IDS for NSL-KDD PKID using HNB-SVM model

Chapter 5

Conclusion and Future Work

In this research, three data mining based network intrusion detection systems are proposed. First, one class classifier is used as anomaly detection method to profile normal connections, and then detect anomalous that deviates from the normal patterns. The one class classifier algorithm combines density and class probability estimation, this algorithm is based on the generation of artificial data from a reference distribution to form a two-class classification. The random forest algorithm is used for class probability estimation and gaussian density as density estimator. Experiments show promising results when the proposed anomaly detection method compared to conventional one class SVM. Secondly, a new hybrid model is proposed to overcome the drawbacks of misuse and anomaly detection methods; where misuse detection cannot detect novel intrusions that are not trained on before, and anomaly detection can detect novel attacks with higher false alarm rate. Our hybrid model combined Hidden Naïve Bayes algorithm as misuse detection algorithm and one class classifier as outlier detection. Two alternatives are used: the one class SVM, and the one class classification algorithm that used in the first model. HNB is used to detect known attacks, then the normally classified instances are evaluated using anomaly detection model, which profiles normal instances on learning step. Different combinations are built for the hybrid model with EMD and PKID discretized datasets. Experiments show that hybrid models usually outperforms or at least similar performance of anomaly and misuse detection models; and HNB-OCC hybrid model is the leading one in terms of DR, FAR and time consumption.

The last model is an improvement of the second model; an additional layer of clustering is added to anomaly detection phase to improve the proposed hybrid intrusion detection system. Decomposing data to smaller homogeneous groups using clustering makes the pattern building fast and easy since each model focused on its own data, and this was alleviated the effect of changing parameters. Results show that the improved model can reach better DR and FAR than hybrid model despite difficulties faced in tuning parameters.

Our experiments show that EMD discretization performed better with OCC-RF algorithm rather than PKID discretization algorithm. Also models that used OCC are faster than models used one-class SVM algorithm and this is a critical issue in many systems.

KDDCup and NSL-KDD datasets has 41 features; updating the proposed hybrid model using feature selection algorithms to choose best features set related with different model is still wider field for our future work.

Tuning parameters for OCC-RF and one-class SVM are not the easy work; automatically choosing parameters for the proposed intrusion detection models need more experiments and remain for the future work.

For our improved hybrid model; k-means is used with k equals to ten, we consider ten clusters for testing and proving the theory since it is may be not the optimized number of clusters. In the future, more testing is needed for number of clusters appropriate for

each dataset. Using and testing other clustering algorithms to decompose the data is still needed, such as density-based clustering DBSCAN and grid clustering, etc.

We can apply our proposed models in real life IDS. Hybrid model can be applied directly, we need labeled database of real connection contains normal and attack connections, tuning the parameters of (OCC-RF or one-class SVM) can be optimized by security officer according to data and the nature of the business. If the business is critical, then security officer may adapt parameters to build narrow hyper-plan therefore DR is high and FAR is also high; alarms can be reviewed by security officer later. Else if the degree of the sensitivity of the business is normal and speed is needed, then wider hyper-plan will be suitable.

For improved hybrid model using k-means clustering, clusters may be considered as categories for IDS (normal, attack family1, attack family2, and so on), parameters that control each cluster (SVM or OCC) are by default optimized according to test set by the system developer, the security officer optimize these clusters to meet his business needs.

References

- [1] C. Azad and V. K. Jha, "Data Mining in Intrusion Detection: A Comparative Study of Methods, Types and Data Sets," *Int. J. Inf. Technol. Comput. Sci.*, vol. 5, no. 8, pp. 75–90, Jul. 2013.
- [2] C. Computer Security Inst., San Francisco, "CSI/FBI Computer Crime and Security Survey," 2004.
- [3] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely, and M. M. Fahmy, "A hybrid network intrusion detection framework based on random forests and weighted k-means," *Ain Shams Eng. J.*, 2013.
- [4] M. Zulkernine and A. Haque, "Random-Forests-Based Network Intrusion Detection Systems," *IEEE Trans. Syst. Man. Cybern.*, vol. 38, no. 5, pp. 649–659, 2008.
- [5] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied Soft Computing*. 2010.
- [6] D. E. Denning, "An Intrusion-Detection Model," *IEEE Trans. Softw. Eng.*, vol. SE-13, 1987.
- [7] G. Kim, S. Lee, and S. Kim, "Expert Systems with Applications A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," 2013.
- [8] C. J. Cole E, Krutz R, *Network Security Bible*. Wiley Publishing, Inc., 2005.
- [9] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert Syst. Appl.*, vol. 29, pp. 713–722, 2005.
- [10] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks.," in *LISA '99: 13th Systems Administration Conference*, 1999, pp. 229–238.
- [11] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, 2005, vol. 38, pp. 333–342.
- [12] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," in *Applications of data mining in computer security*, 2002, pp. 77–101.
- [13] S.-J. Horng, M.-Y. Su, Y.-H. Chen, T.-W. Kao, R.-J. Chen, J.-L. Lai, and C. D. Perkasa, "A novel intrusion detection system based on hierarchical clustering and support vector machines," *Expert Syst. Appl.*, vol. 38, no. 1, pp. 306–313, Jan. 2011.

- [14] U. Aickelin and S. Cayzer, "The Danger Theory and Its Application to Artificial Immune Systems," in *1st International Conference on Artificial Immune Systems ICARIS*, 2008, pp. 141–148.
- [15] H. Liu and H. Motoda, "Computational Methods of Feature Selection," *Zhurnal Eksp. i Teor. Fiz.*, 2007.
- [16] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28. pp. 18–28, 2009.
- [17] S. V Sabnani, "Computer Security : A Machine Learning Approach Computer Security : A Machine Learning Approach," *Mach. Learn.*, no. January, 2008.
- [18] R. L. Krutz and J. Conley, *Wiley Pathways Network Security Fundamentals*. John Wiley & Sons, 2007, p. 524.
- [19] M. Bishop, *Introduction to Computer Security*. Prentice Hall PTR, 2004, p. 784.
- [20] Sushil Jajodia, *Network Intrusion Detection and Prevention Advances in Information Security*. 2010.
- [21] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*, 2009.
- [22] Microsoft, "Microsoft's STRIDE threat model." 2002.
- [23] D. D. Clark and D. R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *NIST Spec. Publ. SP*, vol. 0, pp. 184–194, 1987.
- [24] {Department of Defense}, "Trusted computer system evaluation criteria," *{Department of Defense}*, pp. 1–116, 1985.
- [25] M. Maloof, *Machine learning and data mining for computer security*. Springer, 2006.
- [26] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Comput. Networks*, vol. 31, pp. 805–822, 1999.
- [27] W. Stallings, "Network Security Essentials: Applications and Standards (3rd Edition)," Jul. 2006.
- [28] Y. Wang, "Statistical Opportunities, Roles, and Challenges in Network Security," in *Information Science Reference*, 2008.
- [29] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, and M. Embrechts, "Network-Based Intrusion Detection Using Neural Networks," *Intell. Eng. Syst. through Artif. Neural Networks*, vol. 12, pp. 579–584, 2002.

- [30] K. Leung and C. Leckie, “Unsupervised anomaly detection in network intrusion detection using clusters,” *Proc. Twenty-eighth Australas. ...*, vol. 38, no. January, 2005.
- [31] K. Hempstalk, E. Frank, and I. H. Witten, “One-class classification by combining density and class probability estimation,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5211 LNAI, pp. 505–519.
- [32] W. Hu and Y. Liao, “Robust support vector machines for anomaly detection in computer security,” in *Proc. 2003 International Conference on*, 2003.
- [33] R.-C. Chen, K.-F. Cheng, and C.-F. Hsieh, “Using Rough Set and Support Vector Machine for Network Intrusion Detection,” *Netw. Secur.*, vol. 1, p. 13, 2010.
- [34] A. M. Chandrasekhar and K. Raghuvver, “Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers,” in *2013 International Conference on Computer Communication and Informatics*, 2013, pp. 1–7.
- [35] and M. Z. R. Dewan Md. Farid, Nouria Harbi, “combining naive bayes and decision tree for adaptive intrusion detection,” *Int. J. Netw. Secur. Its Appl.*, vol. volume 2, 2010.
- [36] L. Koc, T. A. Mazzuchi, and S. Sarkani, “A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier,” *Expert Syst. Appl.*, vol. 39, pp. 13492–13500, 2012.
- [37] and B. S. R. Smith, A. Bivens, M. Embrechts, C. Palagiri, “Clustering approaches for anomaly based intrusion detection,” 2002.
- [38] S. M. Bridges and R. B. Vaughn, “Fuzzy data mining and genetic algorithms applied to intrusion detection,” *Proc. 23rd Natl. Inf. Syst. Secur. Conf. held Balt. MA Oct. 1619 2000*, pp. 13–31, 2000.
- [39] and X. L. Q. Tran, H. Duan, “One-class support vector machine for anomaly network traffic detection,” *Work. 18th APAN, Cairns*, 2004.
- [40] B. Agarwal and N. Mittal, “Hybrid Approach for Detection of Anomaly Network Traffic using Data Mining Techniques,” *Procedia Technol.*, vol. 6, pp. 996–1003, 2012.
- [41] S. S. Sivatha Sindhu, S. Geetha, and a. Kannan, “Decision tree based light weight intrusion detection using a wrapper approach,” *Expert Syst. Appl.*, vol. 39, no. 1, pp. 129–141, Jan. 2012.
- [42] D. Barbará, J. Couto, S. Jajodia, L. Popyack, and N. Wu, “ADAM: Detecting Intrusions by Data Mining,” in *IN PROCEEDINGS OF THE IEEE*

- WORKSHOP ON INFORMATION ASSURANCE AND SECURITY*, 2001, pp. 11–16.
- [43] D. Anderson and T. Frivold, “Next-generation intrusion detection expert system (NIDES): A summary,” 1995.
- [44] Z. Muda, W. Yassin, M. N. Sulaiman, and N. I. Udzir, “Intrusion detection based on K-means clustering and OneR classification,” in *Proceedings of the 2011 7th International Conference on Information Assurance and Security, IAS 2011*, 2011, pp. 192–197.
- [45] S. Thaseen and C. a. Kumar, “An analysis of supervised tree based classifiers for intrusion detection system,” *2013 Int. Conf. Pattern Recognition, Informatics Mob. Eng.*, pp. 294–299, Feb. 2013.
- [46] N. Sharma and S. Mukherjee, “A Novel Multi-Classifer Layered Approach to Improve Minority Attack Detection in IDS,” *Procedia Technol.*, vol. 6, pp. 913–921, 2012.
- [47] K. Irani and U. Fayyad, “Multi-Interval Discretization of Continuous-Valued Attributes for Classification learning,” *Proc. Natl. Acad. Sci. U. S. A.*, pp. 1022–1027, 1993.
- [48] H. Liu, F. Hussain, C. L. Tan, and M. Dash, “Discretization: An enabling technique,” *Data Min. Knowl. Discov.*, vol. 6, pp. 393–423, 2002.
- [49] L. Jiang, H. Zhang, and Z. Cai, “A novel bayes model: Hidden naive bayes,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, pp. 1361–1371, 2009.
- [50] Y. Yang and G. I. Webb, “A Comparative Study of Discretization Methods for Naive-Bayes Classifiers,” *Knowl. Acquis.*, vol. 2002, pp. 159–173, 2002.
- [51] J. Dougherty, R. Kohavi, and M. Sahami, “Supervised and unsupervised discretization of continuous features,” *Mach. Learn. Proc. Twelfth Int. Conf.*, vol. 54, pp. 194–202, 1995.
- [52] V. Bolón-Canedo, N. Sánchez-Marroño, and A. Alonso-Betanzos, “A combination of discretization and filter methods for improving classification performance in KDD Cup 99 dataset,” in *Proceedings of the International Joint Conference on Neural Networks*, 2009, pp. 359–366.
- [53] V. Bolón-Canedo, N. Sánchez-Marroño, and A. Alonso-Betanzos, “Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset,” *Expert Syst. Appl.*, vol. 38, pp. 5947–5957, 2011.
- [54] Y. Yang and G. I. Webb, “Weighted proportional k-interval discretization for naive-Bayes classifiers,” *Proc. PAKDD*, pp. 501–512, 2003.
- [55] H. Zhang and S. Sheng, “Learning weighted naive bayes with accurate ranking,” in *Proceedings - Fourth IEEE International Conference on Data Mining, ICDM 2004*, 2004, pp. 567–570.

- [56] R. Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996, vol. 7, pp. 202–207.
- [57] K. Hempstalk and E. Frank, "Discriminating Against New Classes : One-Class versus Multi-Class Classification." .
- [58] L. (University of C. Breiman, *Random forest*, vol. 45. 1999, pp. 1–35.
- [59] D. Gao, Y. Zhang, and Y. Zhao, "Random forest algorithm for classification of multiwavelength data," *Research in Astronomy and Astrophysics*, vol. 9, no. 2. pp. 220–226, 2009.
- [60] A. Liaw and M. Wiener, "Classification and Regression by randomForest," *R news*, vol. 2, pp. 18–22, 2002.
- [61] R. Diaz-Uriarte and S. A. de Andres, "Variable selection from random forests: application to gene expression data," pp. 1–11, Mar. 2005.
- [62] J. Albert, E. Aliu, and H. Anderhub, "Implementation of the Random Forest method for the Imaging Atmospheric Cherenkov Telescope MAGIC," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 588, no. February, pp. 424–432, Sep. 2008.
- [63] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7376 LNAI, pp. 154–168.
- [64] L. M. Manevitz, "One-Class SVMs for Document Classification," *J. Mach. Learn. Res.*, vol. 2, pp. 139–154, 2001.
- [65] L. Zhuang and H. Dai, "Parameter optimization of kernel-based one-class classifier on imbalance learning," *J. Comput.*, vol. 1, no. 7, pp. 32–40, 2006.
- [66] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines—an application in machine fault detection and classification," *Computers & Industrial Engineering*, vol. 48. pp. 395–408, 2005.
- [67] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution.," *Neural Comput.*, vol. 13, pp. 1443–1471, 2001.
- [68] K. Singh, D. Malik, and N. Sharma, "Evolving limitations in K-means algorithm in data mining and their removal," *Int. J. Comput. ...*, vol. 12, no. April, pp. 105–109, 2011.
- [69] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

- [70] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques (Google eBook)*. 2011, p. 664.