A STATISTICAL PROCESS CONTROL APPROACH FOR NETWORK INTRUSION DETECTION

A Dissertation
Presented to
The Academic Faculty

By

Yongro Park

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
May 2005

A STATISTICAL PROCESS CONTROL APPROACH FOR NETWORK INTRUSION DETECTION

Approved by:

Dr. Kwok-Leung Tsui, Advisor
School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. Seong-Hee Kim, Co-advisor
School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. David M. Goldsman
School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. Paul M. Griffin
School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. Wenke Lee
College of Computing
Georgia Institute of Technology

Date Approved: January 2005

In memory of father and brother

ACKNOWLEDGEMENTS

I would like to express full of my sincere gratitude to my advisors, Professor Kwok-Leung Tsui and Assistant Professor Seong-Hee Kim. They were always willing to provide advice, academic and otherwise to help me along my way. I am very grateful to Professor Tsui not only for his invaluable guidance, patience and support throughout my Ph.D. study in GaTech, but also for his great care and help during my staying in USA. I also would like to thank Prof. Kim for her useful advice and consistent support for my research.

I also wish to express my grateful appreciation to all faculty members in department of Industrial and Systems Engineering at Georgia Tech, especially Professor David M. Goldsman, Professor Paul M. Griffin. And I wish to express my appreciation to Professor Wenke Lee in College of Computing at Georgia Tech, too. I can never forget their kind assistances in my Ph.D. study.

I extend all of my gratitude to my friends in Georgia Tech who provided a lot of help during my research and made my life colorful. Although I could not list all of their names here, their warmth would live in my heart all over the time.

To my wife Minsun, without her support I could not be a doctor. And thank you my mom, JungKi for her full support of both mental and financial areas. Finally, I want to dedicate this thesis to my late father, Yong Man Park who always called me doctor Park in his lifetime.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Intrusion detection systems (IDS) have a vital role in protecting computer networks and information systems. In this thesis we applied an SPC monitoring concept to a certain type of traffic data in order to detect a network intrusion.

We developed a general SPC intrusion detection approach and described it and the source and the preparation of data used in this thesis. We extracted sample data sets that represent various situations (e.g., idle/busy, attack/no attack), calculated event intensities for each situation, and stored these sample data sets in the data repository for use in future research.

A regular batch mean chart was used to remove the sample data's inherent 60-second cycles. However, this proved too slow in detecting a signal because the regular batch mean chart only monitored the statistic at the end of the batch. To gain faster results, a modified batch mean (MBM) chart was developed that met this goal. Subsequently, we developed the Modified Batch Mean Shewhart chart, the Modified Batch Mean Cusum chart, and the Modified Batch Mean EWMA chart and analyzed the performances of each one on simulated data. The simulation studies showed that the MBM charts perform especially well with large signals — the type of signal typically associated with a DOS intrusion.

The MBM Charts can be applied two ways: by using actual control limits or by using robust control limits. The actual control limits must be determined by simulation, but the robust control limits require nothing more than the use of the recommended limits. The robust MBM Shewhart chart was developed based on choosing appropriate values based

on batch size. The robust MBM Cusum chart and robust MBM EWMA chart were developed on choosing appropriate values of charting parameters.

In conclusion, (1) sample data sets were developed for future research related to IDS; (2) general guidelines were proposed for applying SPC methods to the type of data typically encountered in IDS, such as how to preprocess the raw data to make it satisfy certain assumptions and how to determine actual and recommended control limits for SPC charts; and (3) the modified batch mean (MBM) concept was developed, creating a technique that can be embedded in various SPC charts to aid them in the detection of a large signal (DOS attack) earlier than is possible with regular SPC charts that rely on batch means.

CHAPTER 1

INTRODUCTION

We now live in the information age. It is nearly impossible to imagine our lives without the Internet and information systems. Nowadays the Internet is used routinely for stock trading, access to weather forecasts and even daily newspapers. The networking revolution has fully come of age in the last decade. More than ever before, we see how the Internet is changing the way humans live. While the possibilities and opportunities afforded by computer information systems are steadily expanding, so too is the risk of malicious intrusions, such as computer viruses or the theft of data.

In this chapter, we discuss the motivation for the need to secure computer network information systems and the role of intrusion detection within this security requirement. In the first section, we define computer system security and common threats to the system. In the second and the third sections, respectively, we define intrusion detection and outline a few popular approaches. Section 4 defines important terminology used throughout the thesis, and Section 5 outlines the structure of the thesis.

In this chapter, we also give a broad overview of the field of intrusion detection as it is presented in the literature. In the next chapter we will survey approaches that have been taken for detecting intrusions.

1.1 COMPUTER SYSTEM SECURITY

Garfinkel and Spafford (1991) define a secure computer system as one that can be depended upon to behave as expected. The integrity that is displayed between the expected behavior and the exhibited behavior is referred to as trust in the security of the computer system. They define the level of trust as an indication of the confidence in the expected behavior of the computer system. The expected behavior is incorporated into the security policy of the computer system and governs the goals that the system must meet.

Russel and Gangemi (1991) introduce a narrower definition of computer security based on the realization of confidentiality, integrity, and availability in a computer system. They define *confidentiality* to mean that information is accessible only to those authorized to access it; *integrity* assures that information remains unaltered by accident or malicious tampering; and *availability* ensures that the computer system remains working when needed without degradation of access to authorized users.

Kumar (1995) defines a secure computer system as a system that protects its data and resources from unauthorized access, tampering, and denial of service. In his framework, data confidentiality is important to commercial success and national security, data integrity allows a hospital to maintain patients' medical histories in order to make critical life decisions, and data availability permits on-line trading in real time.

Threats to Security

We live in a society in which we are increasingly dependent on rapid access to and processing of information. As this demand increases, more information is being stored on computer systems that make possible the rapid tabulation of data from different sources.

The correlation of information from different sources has allowed additional information to be inferred that may be difficult to obtain directly. However, the proliferation of inexpensive computers and networks has exacerbated the problem of unauthorized access to data and tampering with it. (Kumar, 1995)

Increased connectivity not only facilitates access to larger amounts and more varied data than ever before, but also it provides an access path to the data from virtually anywhere on the network (Power, 1995). In many cases, such as in the Internet worm attack of 1988 (Spafford, 1989), network intruders easily overcome the password authentication mechanisms designed to protect these systems.

With an increased understanding of how systems work, intruders have become skilled at determining weaknesses in systems and exploiting them to obtain privileges that allow them to wreak havoc (Kumar 1995). Intruders also use patterns of intrusion that are difficult to trace and are adept at preventing discovery of their identities. They skillfully cover their tracks so that their activity on the penetrated system is not easily discovered. So the threats to computer system security are increasingly intelligent, elusive and destructive.

Detecting these Threats

Lampton (1974) states that most computer systems provide an access control mechanism as their first line of defense. However, this usually only limits access to an object in the system rather than restricting what a subject may do once access has been obtained (Dennis, 1982). Therefore, access control does not necessarily prevent unauthorized information flow into or out of the system once the intruder has gained access to the system's objects (Kumar, 1995).

Information flow can be controlled to enhance security by applying models such as the Bell and LaPadula model (Bell, 1973) to provide secrecy, or the Biba model (Biba, 1977) to provide integrity. However, this enhanced security comes at the price of greatly curtailed convenience. Both models restrict read and write operations to ensure that confidentiality and integrity of data in the system cannot be compromised. If both models are jointly used, the resulting model only permits access to objects at the same security classification level as the subject. This may result in a completely secure system but one of limited utility.

Kumar (1995) notices that access control and protection models are useless against insider threats or compromise of the authentication module. If a password is compromised, access control measures cannot prevent the loss or corruption of information that the user of the compromised password was authorized to access. In general, static methods of computer and network security such as access control and protection models may simply be insufficient to achieve their goal or they may be overly restrictive to users. For example, static techniques may not prevent a violation of security policy such as the browsing of data files, and mandatory access controls that require an appropriate clearance make the system cumbersome to use. Therefore, a dynamic method, such as behavior tracking, is needed to detect and prevent breaches in security.

Intrusion detection systems that perform this role usually form the last line of defense in the overall protection scheme of a computer system. They are useful not only in detecting breaches of security, but also in monitoring attempts so as to provide timely information for countermoves.

1.2 WHAT IS INTRUSION DETECTION?

Anderson (1980) introduced the concept of intrusion detection in 1980 and defined the term "threat" or "intrusion attack" as the potential possibility of a deliberate unauthorized attempt to

- Access information,
- Manipulate information, or
- Render a system unreliable or unusable.

Heady, Luger, and Maccabe (1990) define an intrusion as any set of actions that attempts to compromise the integrity, confidentiality, or availability of a resource. And Kumar (1995) defines an intrusion as a violation of the security policy of the system.

Anderson (1980) also classifies intruders into two types, *external* intruders who are unauthorized users of the machines they attack, and *internal* intruders who have permission to access select portions of the system. He further categorizes internal intruders into those who *masquerade* as another user, those with *legitimate* access to sensitive data, and the most dangerous type, the *clandestine* intruders who have the power to turn off audit control for themselves.

Detection of intrusions is generally divided into two categories: **anomaly** intrusion detection and **misuse** intrusion detection (Deming, 1987). The first refers to intrusions that can be detected based on anomalous behavior and use of computer resources. For example, if user X only uses the database computer from his office, a remote login session activity on this account late at night is anomalous and hence might be an intrusion. As illustrated by this example, anomaly detection attempts to quantify the usual or acceptable behavior and flags irregular behavior as potentially intrusive.

One of the earliest reports that outlines how intrusion may be detected by identifying "abnormal" behavior is the work done by Anderson (1980). In his report, Anderson (1980) presents a threat model that classifies a threat as one of three types. These are external penetrations, internal penetrations, and misfeasance. He uses these classifications to develop a surveillance system based on detection anomalies in user behavior. External penetrations are defined as intrusions that are carried out by unauthorized computer system users; internal penetrations are those that are carried out by authorized users of computer systems who are not authorized for the data that is compromised; and misfeasance is defined as misuse of authorized data and other resources by otherwise authorized users.

Misuse detection refers to intrusions that follow well-defined patterns of attack that exploit weaknesses in system and application software (Kumar, 1995). Such patterns can be precisely written in advance. For example, exploitation of the *finger* and *sendmail* used in the Internet Worm attack (Spafford, 1989) would come under this category. This technique represents knowledge about bad or unacceptable behavior and seeks to detect it directly, as opposed to anomaly intrusion detection, which seeks to detect the complement of normal behavior (Smaha, 1992).

The aforementioned schemes of classifying intrusions as anomaly detection or as misuse detection differ in their methods of detection. Smaha (1988) presents another classification scheme based on intrusion types. He classified intrusions into the following six types:

1. **Attempted break-in:** often detected by profiles of atypical behavior or violations of security constraints.

2. **Masquerade attack:** often detected by profiles of atypical behavior or violations of security constraints.
3. **Penetration of the security control system**: usually detected by monitoring for specific patterns of activity.
4. **Leakage**: often detected by atypical usage of Input/Output (I/O) resources.
5. **Denial of service**: often detected by atypical usage of system resources.
6. **Malicious use**: often detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

Although there are many ways to classify an intrusion, the main techniques used for detecting intrusions are the same: the statistical approach of anomaly detection, and the precise monitoring of known attack methods in the misuse detection approach. It is necessary to understand that both approaches have implicit and crucial assumptions about the nature of intrusions they can detect.

1.3 THE NEED FOR INTRUSION DETECTION SYSTEMS

A computer system should provide *confidentiality*, *integrity* and *availability* against denial of service. However, because of increased connectivity (especially on the Internet), and the vast spectrum of financial possibilities that are opening up, more and more systems are subject to attack by intruders. These attempts at subversion try to exploit flaws in the operating system as well as in application programs and have resulted in spectacular incidents like the Internet Worm incident of 1988 (Spaffod, 1989).

There are two ways to handle subversion attempts. One way is to prevent subversion itself by building a completely secure system. We could, for example, *require* all users to

identify and authenticate themselves; we could protect data by various cryptographic methods and very tight access control mechanisms. However this is not really feasible.

Thus, we are stuck with systems that have vulnerabilities. If there are attacks on a system, we would like to detect them as soon as possible (preferably in real-time) and take appropriate action. This is essentially what an intrusion detection system (IDS) does.

An IDS is defined as a system that detects intrusion. An IDS does not usually take preventive measures when an attack is detected; it is a reactive rather than pro-active agent. It plays the role of an informant rather than a police officer.

The most popular way to detect intrusions has been by using the *audit data* generated by the operating system. An *audit trail* is a record of activities on a system that is logged to a file in chronologically sorted order. Since almost all activities are logged on a system, it is possible that a manual inspection of these logs would allow intrusions to be detected. However, the incredibly large size of the audit data generated makes manual analysis impossible. IDSs automate the drudgery of wading through the jungle of audit data. Audit trails are particularly useful because they can be used to establish the guilt of attackers, and they are often the only way to detect unauthorized but subversive user activity.

Many times, even after an attack has occurred, it is important to analyze the audit data for several reasons. These include determining the extent of damage, tracking down the attackers, and taking steps to prevent future attacks. An IDS can also be used to analyze audit data for such insights. This makes an IDS valuable in real-time as well as a post-mortem analysis tool.

1.4 TERMINOLOGIES

This section explains several terms used throughout the thesis. The terms have well-accepted definitions among security professionals (Kumar, 1995).

Audit record/event. An audit record is an individual entry in an audit trail. It is also referred to in this proposal as an "event." The number of distinct event types is finite and known a priori. Events are tagged with data. There is a "type" field with every event that distinguishes IT? among the different events in the stream of activity. Events can have any number (usually small) of tag fields. The exact number and nature of the fields is dependent on the types of event. The layout of each event is fixed, although each event type can have a different layout.

Audit trail/event stream. An audit trail is defined in Longley (1987) as a chronological record of system activity sufficient for enabling the reconstruction, review and examination of the sequence of environments and activity surrounding or leading to each event in the path of a transaction from inception to output of final results.

The term "event stream," against which signatures are matched, is used in this thesis in the same sense as an audit trail. In practice, audit trails record service requests that applications make of the operating system, and events are recorded when applications make system calls. Recording system service requests from applications provide a trustworthy application independent monitoring technique that works for all applications, without requiring intrusive instrumentation of the applications. Some important

applications such as login have, however, been retrofitted to generate their own specific events that overlap with other events in the audit trail.

BSM (Basic Security Module). The Solaris operating system from Sun Microsystems Inc. has a security extension called the basic security module (BSM). BSM supports the monitoring of activities on a host machine by recording security-relevant events. BSM-auditable events fall into two categories: kernel events and user-level events. Kernel events are generated by system calls to the kernel of the Solaris operation system. User-level events are generated by application software. A BSM audit record for each event contains a variety of information, including the event type, user ID, group ID, process ID, session ID, and the system object accessed, among others.

C2 security rating of computer systems. A Department of Defense security evaluation criteria class requiring audition and protection of encrypted passwords, among others, as described in the Orange Book (DoDS, 1985). The primary motivation behind the Orange Book was the need to quantify security and trust because different organizations and different types of information require different types of security (Russell, 1991). C2 is a category of a specific set of criteria to be met by computer systems defined by the Orange Book (DoDs, 1985).

DOS (Denial of Service). When a denial of service (DOS) attack occurs, a computer or a network user is unable to access resources like e-mail and the Internet. An attack can be directed at an operating system or at the network. Denial of service attacks can happen by

flooding the network with excessive traffic. The network is unable to distinguish between legitimate traffic and malicious traffic during the attack.

Other kinds of DOS attacks include using up all the victim's bandwidth instead of targeting a particular service or by using all of a system's resources, like memory, on a server. For example, attackers could also try to shut down a system by flooding the network with e-mail.

Exploitation. An exploitation is a set of actions that result in a violation of the security policy of a computer system. Intruders exploit system vulnerabilities or flaws to gain unauthorized access to the system. These exploitations can often be encoded as signatures that can be matched against the audit trail for detection.

Flaw. A flaw is defined in Longley (1987) as an error of commission, omission or oversight in a system that allows protective mechanisms to be bypassed. In this thesis, we use the terms vulnerabilities and flaws synonymously.

Signature. In detection of misuse intrusion, a signature is the specification of features, conditions, arrangements and interrelationships among events that signify a break-in or other misuse, or their attempt. The term "pattern" and "intrusion pattern" are used throughout the thesis in the same sense as a signature.

Spoofing. A spoofing is the creation of TCP/IP packets using somebody else's IP address. Routers use the "destination IP" address in order to forward packets through the Internet,

but ignore the "source IP" address that is only used by the destination machine when it responds back to the source. IP spoofing is an integral part of many network attacks that do not need to see responses (blind spoofing).

TCP/IP (Transmission Control Protocol/Internet Protocol). TCP/IP is developed by a Department of Defense (DOD) research project to connect a number different networks designed by different vendors into a network of networks. TCP/IP uses several protocols, the two main ones being TCP and IP. TCP/IP is built into the UNIX operating system and is used by the Internet, making it the de facto standard for transmitting data over networks. Even network operating systems that have their own protocols, such as Netware, also support TCP/IP.

Vulnerability. A vulnerability is defined in Longley (1987) as a weakness in automated system security procedures, administrative controls or internal controls that could be exploited to gain unauthorized access or to disrupt critical processing. Anderson (1980) defines vulnerability in a less abstract way as a known or suspected flaw in the hardware or software design or operation of a system that exposes it to penetration of its information.

1.5 ORGANIZATION OF THE THESIS

In this thesis, we are concerned with the SPC charting methods in the computer network intrusion detection areas. We stress that we are interested in DOS intrusion attacks that can be detected with activity monitoring schemes. Recognizing the effectiveness of the control charts for the DOS intrusion attack, we build a SPC intrusion detection approach, and propose new SPC charts that allow us to detect DOS intrusion signals faster and more effectively than other SPC charts now in use. This new family of charts is named Modified Batch Mean (MBM) charts. The new charts are analyzed and compared with existing charting methods using the simulated data.

Below is an outline of the thesis.

Chapter 1 introduces the problem of computer security and the need for intrusion detection systems and the associated terminologies that will be used in the thesis.

Chapter 2 reviews literatures dealing with intrusion detection systems and research.

Chapter 3 reviews SPC methods widely used in manufacturing and process control and describes the performance measures of the charts.

Chapter 4 studies the SPC approach for the intrusion detection. The procedures and the results of the case study with the procedure are presented.

Chapter 5 presents the characteristics of the input sample data and the simulation modeling for the SPC intrusion detection approach.

Chapter 6 illustrates new SPC methods presented in the thesis and the results of the simulation using the new SPC methods are also presented.

Chapter 7 presents the results of applying the new SPC methods to the data used in the case study.

Chapter 8 provides a summary of this thesis and a discussion of future directions for research.

CHAPTER 2

INTRUSION DETECTION SYSTEMS

This chapter describes several existing intrusion detection systems. We also describe the data-mining model of intrusion detection proposed by Lee (1999) and the statistical methods proposed by Ye (2002).

2.1 INTRODUCTION

IDSs have three characteristics, namely the audit source, the methods employed, and the response initiated by detection. For intrusion detection systems, the input is the audit source and the process is the mechanism of detection and the output is the system response. Based on its source, an IDS is categorized as either a host-based system or a network-based system. A host-based system generally uses host-based input data and prevents an intrusion into the host. A network-based system gathers information from the network traffic. Based on its response to the information it gathers, an IDS is categorized as either a passive system or active system.

Based on the detection method it employs, an IDS is categorized as a knowledge-based system or a behavior-based system. A knowledge-based system is in other words a misuse detection system and a behavior-based system is an anomaly detection system. This chapter uses these two categories — misuse detection systems and anomaly detection systems — to describe existing IDSs because together they encompass the

techniques of intrusion detection. In the following sections, we will describe anomaly detection systems, misuse detection systems and other methods.

2.2 ANOMALY DETECTION SYSTEMS

Anomaly detection techniques assume that all intrusive activities are necessarily anomalous. This means that if we could establish a "normal activity profile" for a system, we could, in theory, flag all system states varying from the established profile by statistically significant amounts as an intrusion is in progress. If we assume that the set of intrusive activities only intersects the set of anomalous activities instead of being exactly the same, we find several interesting possibilities: (1) Anomalous activities that are not intrusive are flagged as intrusive, i.e., false positives. (2) Intrusive activities that are not flagged as intrusive when they actually are, i.e., false negatives. False negatives are far more serious than false positives.

The main issues in anomaly detection systems thus become the selection of threshold levels so that neither false positives nor false negatives are unreasonably probable and the selection of features that will be monitored. Anomaly detection systems are also computationally expensive because of the overhead involved in keeping track of and updating several metrics of system profile. There have been a few major approaches to anomaly intrusion detection systems, some of which are described below.

Statistical Approaches

In statistical approaches, behavior profiles for subjects are initially generated as the system continues running, and the anomaly detector constantly compares the variance of

the present profile against that of baseline. We note that in this case there may be several measures that affect the behavior profile such as activity measures, CPU time used, number of network connections in a time period, etc. In some systems, the current profile and the previous profile are merged at regular time intervals, but in some other systems profile generation is a one-time activity.

The main advantage of statistical systems is that they adaptively study use behaviors and are thus potentially more sensitive than human experts. However, there are a few problems with statistical approaches. One problem is that intruders can systematically train them so that eventually intrusive events are considered normal. A second is that either false positives or false negatives can be generated, depending on whether the threshold is set too low or too high. And a third is that relationships between events can be missed because of the insensitivity of statistical measures to the order of events.

An open issue with statistical approaches in particular, and with anomaly detection systems in general, is the selection of measures to monitor. The subset of all possible measures that accurately predicts intrusive activities is unknown. Static methods of determining these measures are sometimes misleading because of the unique features of a particular system. Thus, it seems that a combination of static and dynamic determination of the set of measures should be done. Some problems associated with this technique have been remedied by other methods, including the method involving *Predictive Pattern Generation*, which takes past events into account when analyzing the data.

Predictive Pattern Generation

This method of intrusion detection tries to predict future events based on events that have already occurred (Teng, 1990). Therefore, we could have a rule

E1 - E2 --> (E3 = 75%, E4 = 15%, E5 = 5%)

This would mean that given that events E1 and E2 have occurred, with E2 occurring after E1, there is an 75% probability that event E3 will follow, a 15% chance that event E4 will follow and a 5% probability that event E5 will follow. The problem with this is that some intrusion scenarios that are not described by the rules will not be flagged as intrusive. Thus, if an event sequence A - B - C exists that is intrusive, but is not listed in the rule-base, it will be classified as unrecognized and thus be undetected. This problem can be partially solved by flagging as intrusions any unknown events (increasing the probability of false positives), or by flagging them as nonintrusive (thus increasing the probability of false negatives). Typically, however, an event is flagged as intrusive if the left side of a rule is matched, but the right side is very deviant statistically from the prediction.

This approach has several advantages. First, rule- based sequential patterns can detect anomalous activities that were difficult to detect with traditional methods. Second, systems built using this model are highly adaptive to changes. This is because low quality patterns are continuously eliminated, finally leaving only higher quality patterns. Third, it is easier to detect users who try to train the system during its learning period. And fourth, anomalous activities can be detected and reported within seconds of receiving audit events.

Neural Networks

The idea of a neural network is to train it to predict a user's next action or command, given the window of previous actions or commands. The network is trained on a set of representative user commands. After the training period, the network tries to match actual commands with the actual user profile already present in the network. Any incorrectly predicted commands actually measure the deviation of the user from the established profile. Some advantages of using neural networks are that they cope well with noisy data, their success does not depend on any statistical assumption about the nature of the underlying data, and they are easier to modify for new user communities (Lunt, 1993). However, they have some weaknesses. First, a small window will result in false positives while a large window will result in irrelevant data as well as increase the chance of false negatives. Second, the network topology is only determined after considerable trial and error. And third, an intruder can train the network during its learning phase.

2.3 MISUSE DETECTION SYSTEMS

The concept behind misuse detection schemes is that there are ways to represent attacks in the form of a pattern or a signature so that even variations of the same attack can be detected. This means that these systems are not unlike virus detection systems — they can detect many or all known attack patterns, but they are of little use for as yet unknown attack methods. An interesting point to note is that anomaly detection systems try to detect the complement of "bad" behavior but misuse detection systems try to recognize known "bad" behavior.

The main issues in misuse detection systems are how to write a signature that encompasses all possible variations of the pertinent attack, and how to write signatures that also do not match nonintrusive activity.

Significant research on misuse detection systems has been undertaken recently, including projects at SRI (Stanford Research Institute), Purdue University and the University of California-Davis. Some of these systems are explained in depth in this section.

Expert Systems

Expert systems are modeled in such a way as to separate the rule-matching phase from the action phase. The matching is done according to audit trail events. The Next Generation Intrusion Detection Expert System (NIDES) developed by SRI is an interesting case study for the expert system approach. NIDES follows a hybrid intrusion detection technique consisting of a misuse detection component as well as an anomaly detection component. The anomaly detector is based on a statistical approach, and it flags events as intrusive if they are largely deviant from the expected behavior. To do this, it builds user profiles based on more than 30 criteria, including CPU and I/O usage, commands used, local network activity, system errors, etc. (Lunt, 1993). These profiles are updated at periodic intervals. The expert system's misuse detection component encodes known intrusion scenarios and attack patterns (bugs in old versions of *sendmail* could be one vulnerability). The rule database can be changed for different systems. One advantage of the NIDES approach is that it has a statistical component as well as an expert system component. This increases the possibility that one system will catch

intrusions missed by the other. Another advantage is the problem's control reasoning is cleanly separated from the formulation of the solution.

There are some drawbacks to the expert system approach, too. For example, the expert system has to be formulated by a security professional, and thus the system is only as strong as the security personnel who programs it (Lunt, 1993). This means that there is a real chance that expert systems can fail to flag intrusions. It is for this reason that NIDES has an anomaly as well as a misuse detection component. These two components are loosely coupled in the sense that for the most part they perform their operations independently. The NIDES system runs on a machine different from the machine(s) to be monitored, which could be unreasonable overhead. Furthermore, additions and deletions of rules from the rule-base must take into account the inter-dependencies between different rules in the rule-base. And there is no recognition of the sequential ordering of data because the various conditions that make up a rule are not recognized as ordered.

Keystroke Monitoring

Keystroke monitoring is a very simple technique that monitors keystrokes for attack patterns. Unfortunately the system has several defects — features of shells like *bash*, *ksh*, and *tcsh* in which user definable aliases are present defeat the technique unless alias expansion and semantic analysis of the commands is taken up (Kumar, 1995). The method also does not analyze the running of a program, only the keystrokes. This means that a malicious program cannot be flagged for intrusive activities. Operating systems do not offer much support for keystroke capturing, so the keystroke monitor should have a hook that analyses keystrokes before sending them on to their intended receiver. An

improvement to this would be to monitor system calls by application programs as well so that an analysis of the program's execution is possible.

Model-Based Intrusion Detection

Model-based intrusion detection states that certain scenarios are inferred by certain other observable activities. If these activities are monitored, it is possible to find intrusion attempts by looking at activities that infer a certain intrusion scenario. A model-based scheme consists of three important modules of *anticipator*, *planner*, and *interpreter* (Garvey, 1991). The *anticipator* uses active models and the scenario models to try to predict the next step that is expected to occur in the scenario. A scenario model is a knowledge base with specifications of intrusion scenarios. The *planner* then translates this hypothesis into a format that shows the behavior as it would occur in the audit trail. It uses the predicted information to plan what to search for next. The *interpreter* then searches for this data in the audit trail. The system proceeds this way, accumulating more and more evidence of an intrusion attempt until a threshold is crossed; at this point, it signals an intrusion attempt.

This is a very clean approach. Because the planner and the interpreter know what they are searching for at each step, the large amounts of noise present in audit data can be filtered, leading to excellent performance improvements. In addition, the system can predict the attacker's next move based on the intrusion model. These predictions can be used to verify an intrusion hypothesis, to take preventive measures, or to determine what data to look for next.

However, there are some critical issues related to this system. First, patterns for intrusion scenarios must be easily recognized. Second, patterns must always occur in the behavior being looked for. And finally, patterns must be *distinguishing*; they must *not* be associated with any other normal behavior.

State Transition Analysis

In the state transition analysis technique, the monitored system is represented as a state transition diagram. As data is analyzed, the system makes transitions from one state to another. A transition takes place on some Boolean condition being true (for example, the user opening a file). The approach followed in USTAT (State Transition Analysis Tool for UNIX) (Ilgun, 1992) is to have state transitions from safe to unsafe states based on known attack patterns.

There are also a few problems with state transition systems. First, attack patterns can specify only a sequence of events, rather than more complex forms. Second, there are no general-purpose methods to prune the search except through the assertion primitives described above. And finally, they cannot detect denial of service attacks, failed logins, variations from normal usage, and passive listening — this is because these items are either not recorded by the audit trail mechanism, or they cannot be represented by state transition diagrams.

A small point to be noted is that USTAT is never meant to be a stand-alone intrusion detection system; indeed, USTAT is meant to be used with an anomaly detector so that more intrusion attempts may be detected by their combination.

Pattern Matching

Kumar (1995) proposed a new misuse detection system based on pattern matching. This model encodes known intrusion signatures as patterns that are then matched against the audit data. Like the state transition analysis model, this model attempts to match incoming events to the patterns representing intrusion scenarios. The implementation makes transitions on certain events, called *labels*, and Boolean variables called *guards* can be placed at each transition. The difference between this and the state transition model is that the state transition model associates these guards with states, rather than transitions. Kumar (1995) states that the important advantages of this model are:

1. Declarative specification: It only needs to be specified what patterns need to be matched, not how to match them.
2. Multiple event streams can be used together to match against patterns for each stream without the need to combine streams. This means that streams can be processed independently, and their results can be analyzed together to give evidence of intrusive activity.
3. Portability: Since intrusion signatures are written in a system independent script, they need not be rewritten for different audit trails. The patterns' declarative specifications enable them to be exchanged across different operating systems and different audit trails.
4. It has excellent real-time capabilities. Kumar reports a CPU overhead of 5-6% when scanning for 100 different patterns, which is excellent.
5. It can detect some attack signatures, such as the failed logins signature, that the state transition model cannot do.

One problem with this model is it can only detect attacks based on known vulnerabilities (a problem with misuse detection systems in general). In addition, pattern matching is not very useful for representing ill-defined patterns, and it is not an easy task to translate known attack scenarios into patterns that can be used by the model. Also, it cannot detect passive wire-tapping intrusions, nor can in detect spoofing attacks in which a machine pretends to be another machine by using its IP address.

Generic Intrusion Detection Model

Dorothy Denning (1987) introduced a generic intrusion detection model that was independent of any particular system, application environment, system vulnerability, or type of intrusion. The basic idea of the model is to maintain a set of profiles for *subjects* (usually, but not necessarily users of a system). When an audit record is generated, the model matches it with the appropriate profile and then makes decisions on updating the profile, checking for abnormal behavior and reporting anomalies detected. To do this, it monitors system services such as file accesses, executable programs, and logins. It has no specific knowledge of the target system's vulnerabilities, although this knowledge would be extremely useful in making the model more valuable. In fact, the Intrusion Detection Expert System (IDES) developed at SRI was based on this model. The basic ideas in this model appear with little modification in many systems built. However, there are some systems that do not fit easily into this model.

NSM (Network Security Monitor)

NSM is an intrusion detection system developed at the University of California-Davis. NSM is a network-based IDS that differs from all of the IDSs discussed earlier because it does not use or analyze the host machine(s) audit trails. Rather, it monitors network traffic in order to detect intrusions (Mukherjee, 1994).

NSM has several perceived advantages. First, the IDS gets instantaneous access to network data. Second, the IDS is hidden from the intruder because it is passively listening to network traffic. Therefore, it cannot be shut off or its data compromised. Finally, the IDS can be used with any system because it is monitoring network traffic, protocols for which (TCP, UDP etc.) are standardized. For example, there is no problem with different audit files.

Autonomous Agents

Crosbie and Spafford (Crosbie, 1995) proposed to build an IDS using autonomous agents. Instead of a single large IDS defending the system, they propose an approach where several independent, small processes operate while cooperating in maintaining the system. The advantages claimed for this approach are efficiency, fault tolerance, resilience to degradation, extensibility and scalability. The foreseen drawbacks include the overhead of so many processes, long training times, and the fact that if the system is subverted, it becomes a security liability. An interesting possibility they open up is that of an *active* defense, that can respond to intrusions instead of passively reporting them (it could kill suspicious connections, for example).

2.4 OTHER MODELS AND DIRECTIONS IN RESEARCH

Lee and Stolfo (Lee, 1999) propose a framework of applying **Data Mining** techniques to build intrusion detection models. The key ideas are to mine system audit data for consistent and useful patterns of program and user behavior, and use the set of relevant system features presented in the patterns to compute (inductively learned) classifiers that can recognize anomalies and known intrusions. They show that classification models can detect intrusions, provided that sufficient audit data is available for training and the right set of system features are selected by experiments.

This framework consists of classification, association rules and frequent episode programs, as well as a support environment that enables system builders to drive the process of constructing and evaluating detection models interactively and iteratively. The end products are concise and intuitive classification rules (that can be easily inspected and edited by security experts when needed) that can detect intrusions. They also propose that association rules and frequent episodes from the audit data can be used to guide audit data gathering and feature selection, the critical steps in building effective classification models.

Ye, Emran, Chen, and Vilbert (Ye, 2002) investigate a multivariate **Quality Control Technique** to detect intrusions by building a long-term profile of normal activities in information systems (norm profile) and using the norm profile to detect anomalies. They present their work on multivariate statistical analysis of audit trails for host-based intrusion detection. Specifically, Hotelling's T^2 test — a multivariate statistical process control (SPC) technique — is used to analyze audit trails of activities in an information

system and detect host-based intrusions into the information system that leave trails in the audit data. Hotelling's T^2 test is also compared with a more scalable multivariate statistical analysis technique — the chi-squared test.

Ye, Li, Chen, Emran, and Xu (2001) apply a **Markov model** that takes into account the ordering property of multiple events for intrusion detection. They claim that the application of a Markov model helps answer the question about whether the ordering property of activity data provides additional advantage to intrusion detection, or whether one can detect intrusions from only the frequency property of activity data without the ordering property.

Ye, Vilbert, and Chen (2003) also apply the **EWMA** (Exponentially Weighted Moving Average) **Control Chart** method for intrusion detection. They use EWMA statistical control chart methods for detecting intrusive DOS attacks by using the BSM host audit file. They calculate the event intensity by calculating the intensity of the BSM header files and applied the EWMA statistical control charting methods.

CHAPTER 3

STATISTICAL PROCESS CONTROL METHODS

This chapter describes the methods of several common statistical process control charts. The definition of a statistical control chart is that it is a graphical device for monitoring a measurable characteristic of a process for the purpose of showing whether the process is operating within its limits of expected variation. A major objective of a statistical process control chart is "to detect quickly the occurrence of assignable causes of process shifts so that the process can be investigated and corrective action undertaken before many nonconforming units are manufactured" (Montgomery, 2001). The control chart is an on-line process-monitoring technique widely used for this purpose.

3.1 SHEWHART CHART

The most commonly used form of control chart is named after Walter A. Shewhart (1891-1967), who invented it in 1924 and used it as the basis for laying the foundation of modern quality control in his seminal 1931 book, *Economic Control of Quality of Manufactured Product* (Shewhart, 1931).

The basic idea advocated by Shewhart is that there are switches in time that transfer the generating process into a distribution not typical of the dominant distribution. These switches manifest themselves into different average product measurements and variances (Montgomery, 2001).

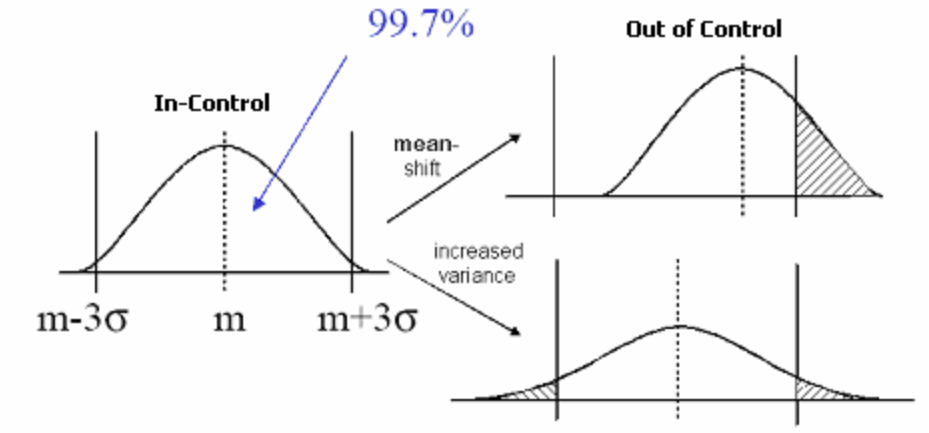


Figure 3.1 In-Control vs. Out of Control

Figure 3.1 shows the in-control and out-of-control states for a process. If a product is to meet customer requirements, it should be produced by a process that is stable or repeatable. More precisely, the process must be capable of operating with little variability from the target or nominal dimensions of the product's quality characteristics. Such a process is considered to be in control. A process with a shift in mean and/or variance shift is considered out of control; such a process is a quality problem and eventually produces defective products that are useless. In the case of a process with normal distribution, if we set the control limit to the 3-s level, the probability of an in-control limit is 99.7 %.

Shewhart defined chance and assignable causes as the two sources of quality variation. Chance is an inherent, inevitable, and unavoidable cause of variation. A process that is operating with only **chance** present a cause of variation is said to be **in statistical control.** The output of a process occasionally has other kinds of variability. This variability in key quality characteristics usually arises from any one or all of three sources: improperly

adjusted or controlled machines, operator errors, or defective raw material. Such variability is generally large when compared to the background noise, and it usually represents an unacceptable level of process performance. Shewhart (1931) refers to these sources of variation that are not the result of chance as "assignable causes." A process that is operating with **assignable causes** is said to be **out of control**.

This brief discussion shows that the underlying concept of a Shewhart chart is to construct its limitations based upon variations allowable as its in-control state and monitor the quality of the product produced. If variations attributable to assignable causes occur, product quality quickly exceeds the control limits and assignable causes can be investigated and corrective action undertaken.

A typical control chart is shown in Figure 3.2, which is a graphical display of a quality characteristic that has been measured or computed from a sample versus the sample number or time. The chart contains a center line that represents the average value of the quality characteristic corresponding to the in-control state. Two other horizontal lines, called the upper control limit (UCL) and the lower control limit (LCL), are also shown on the chart. These control limits are chosen so that if the process is in control, nearly all of the sample points will fall between them. As long as the points plot within the control limits, the process is assumed to be in control.

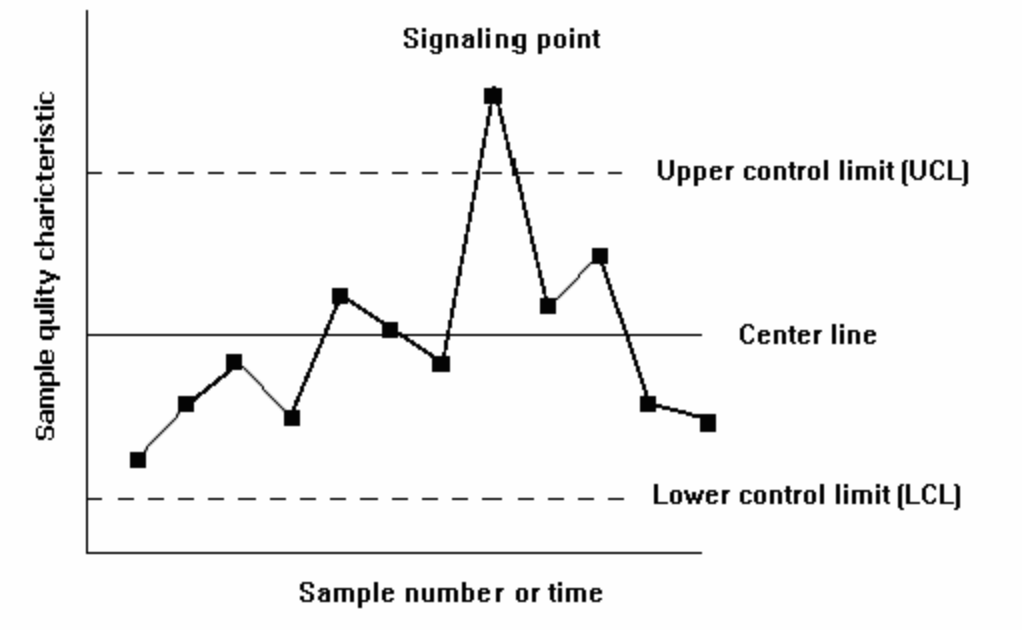


Figure 3.2 A typical Shewhart chart.

The distance of the control limits from the center line is expressed in standard deviation units of monitoring quality statistics. In general, we use 3-s for the distance, and that control limit gives an average run length (ARL) of 370 if the process observations are uncorrelated and have normal distribution. Since the ARL can be calculated by $ARL = \frac{1}{a}$, where a is the probability that any point exceeds the control limits of in-control status. To illustrate, for the Shewhart chart with three-sigma limits, $p = \Pr(|X - m| > 3s) = 0.0027$ is the probability that a single point falls outside the limits when the process is in control, where X follows normal distribution with mean μ and standard deviation s. Therefore, the average run length of the Shewhart chart when the process is in control is

$$ARL = \frac{1}{p} = \frac{1}{0.0027} = 370.$$

3.2 CUSUM CHART

Cumulative sum control charts were first proposed by Page (1961) and have been studied by many authors and are preferred when a small, sustained shift in a process is considered. The basic idea of the Cusum chart is that such small shifts can be easier to see if one "accumulates" deviations from some standard value for Q. The Cusum chart directly incorporates all the information as the sequence of sample values by plotting the cumulative sums of the deviations of the sample value and target value as the following formulation.

$$CUSUM_i = (Q_i - m_0) + CUSUM_{i-1},$$

where Q_i is the quality characteristic objective, m_0 is target value, and $CUSUM_i$ is the i^{th} cumulative sum of quality characteristic .

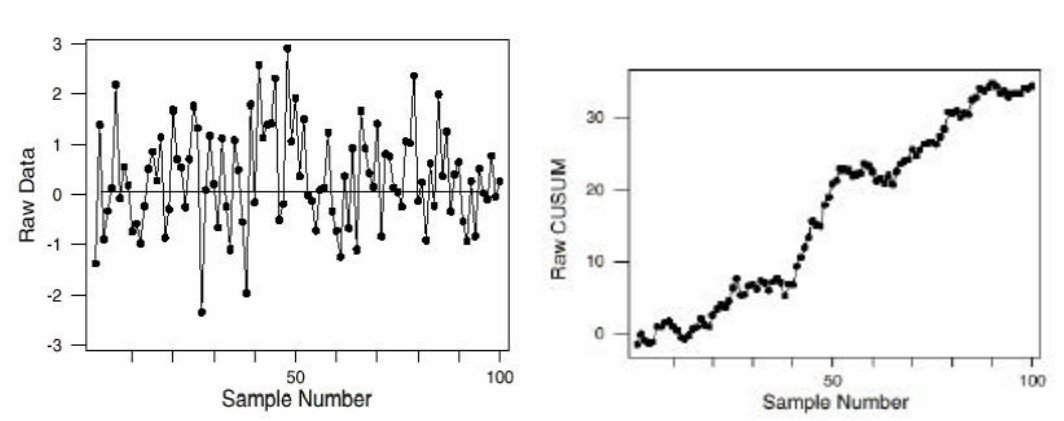


Figure 3.3 Cusum Chart.

Figure 3.3 shows the effectiveness of the Cusum chart in detecting a small mean shift. The left one is a Shewhart chart and the right one is a Cusum chart with an example of a mean shift of 0.5 and a standard deviation of 1 sample case. In the Shewhart chart with a

center line = 0, it is difficult to determine if the mean is greater than 0, but in the Cusum chart in which the mean = 0, the change of mean is apparent. The chart on the right, however, is not a control chart because it lacks statistical control limits.

Cusum charts may be represented in two ways, either as the tabular Cusum or the V-mask form. Montgomery (2001) recommends the use of the tabular form of a Cusum chart because of its convenience. The tabular form has been used in this thesis.

The *V-Mask* is a visual procedure proposed by Barnard in 1959 and sometimes is used to determine whether a process is out of control. A *V*-Mask is an overlay shaped in the form of a *V* lying on its side. The overlay is superimposed on the graph of the cumulative sums. The origin point of the *V*-Mask (see Figure 3.4) is placed on top of the latest cumulative sum point, and past points are examined to see if any fall above or below the sides of the *V*. As long as all the previous points lie between the sides of the *V*, the process is in control. Otherwise (even if only one point lies outside) the process is suspected of being out of control.

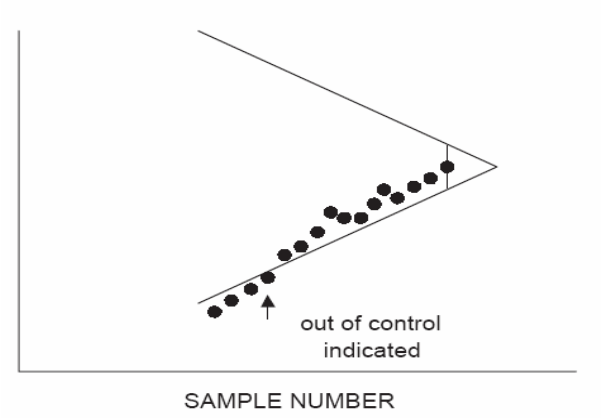


Figure 3.4 V-mask of Cusum chart

The tabular Cusum works by accumulating derivations from m_0 that are above the target with one statistic C^+ and accumulating derivations from m_0 that are below the target with another statistic C^-. The statistics C^+ and C^- are called one-sided upper and lower Cusums, respectively. They are sequentially computed as follows.

$$C_i^+ = \max[0, x_i - (m_0 + K) + C_{i-1}^+], \tag{3.1}$$

$$C_i^- = \max[0, (m_0 + K) - x_i + C_{i-1}^-], \tag{3.2}$$

where x_i is the i^{th} observation on the process and both the starting values are $C_0^+ = C_0^- = 0$. When the process is in control, x_i has a normal distribution with mean m_0 and standard deviation s.

In equation 3.1 and 3.2, K is usually called the reference value and is often chosen about halfway between the target m_0 and the out-of-control values of the mean m_1 that are the target for quick detection. If either C^+ or C^- exceed the decision interval H, the process is considered to be out of control.

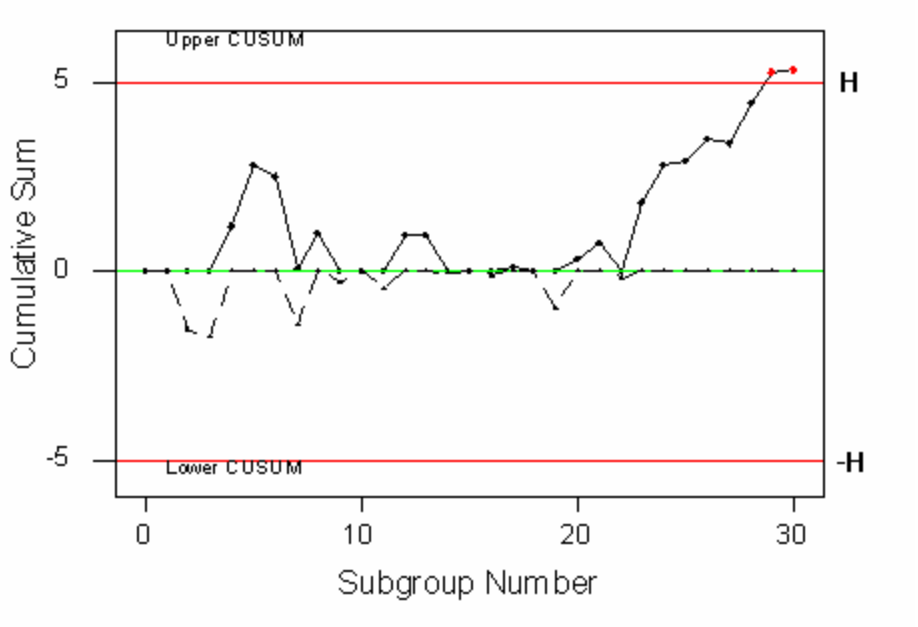


Figure 3.5 Cusum status chart for tabular Cusum

The tabular Cusum is designed by choosing values for the reference value K and the Cusum decision interval H. Montgomery (2001) recommended that these parameters be selected to provide good average run length performance. Many analytical studies of Cusum ARL performance have been undertaken. For example, Hawkins (1993) gives a table of k (=K/s) values and the corresponding h (=H/s) values that will achieve ARL_0 = 370. In his table, if we set k=1/2 then h=4.77 gives us ARL_0 = 370.

3.3 EWMA CHART

The exponential weighted moving average (or EWMA) control chart was introduced by Roberts (1959) and also is a good alternative to the Shewhart control chart for detecting small shifts. The exponentially weighted moving average is defined as

$$z_i = lx_i + (1-l)z_{i-1},\ i = 1, 2, \ldots . \tag{3.3}$$

Where $0 < l \leq 1$ is a constant and the starting value (required with the first sample at i =1) is the process target, so that

$$z_0 = m_0.$$

Sometimes the average of preliminary data is used as the starting value of the EWMA, so that $z_0 = \bar{x}$.

To demonstrate that the EWMA statistic z_i is a weighted average of all previous samples, we may substitute for z_{i-1} on the right side of equation 3.3 to obtain

$$z_i = l x_i + (1 - l)[l x_{i-1} + (1 - l) z_{i-2}]$$

$$= l x_i + l(1 - l) x_{i-1} + (1 - l)^2 z_{i-2}.$$

Continuing to substitute recursively for z_{i-j}, j = 2, 3, …, t, we obtain

$$z_i = l \sum_{j=0}^{i-1} (1-l)^j x_{i-j} + (1-l)^i z_0. \tag{3.4}$$

The weight of a sample decreases geometrically with the age of the sample. Furthermore, the weights sum to unity, since

$$l \sum_{j=0}^{i-1} (1-l)^j = l \left[\frac{1-(1-l)^i}{1-(1-l)} \right] = 1-(1-l)^i.$$

If l =0.2, then the weight assigned to the current sample mean is 0.2, and the weights given to the preceding means are 0.16, 0.128, 0.1024, and so forth. A comparison of these weights with those of a ten-period moving average is shown in Figure 3.6. The blue line represents a case in which l =0.2, and the red line represents a case in which l =0.6. Because these weights decline geometrically when connected by a smooth curve, the

EWMA is sometimes called a geometric moving average (GMA), and if the l value is small, the weights decline faster.

If the observations x_i are independent random variables with variance s^2, then the variance of z_i is

$$s_{z_i}^2 = s^2\left(\frac{l}{2-l}\right)[1-(1-l)^{2i}] \tag{3.5}$$

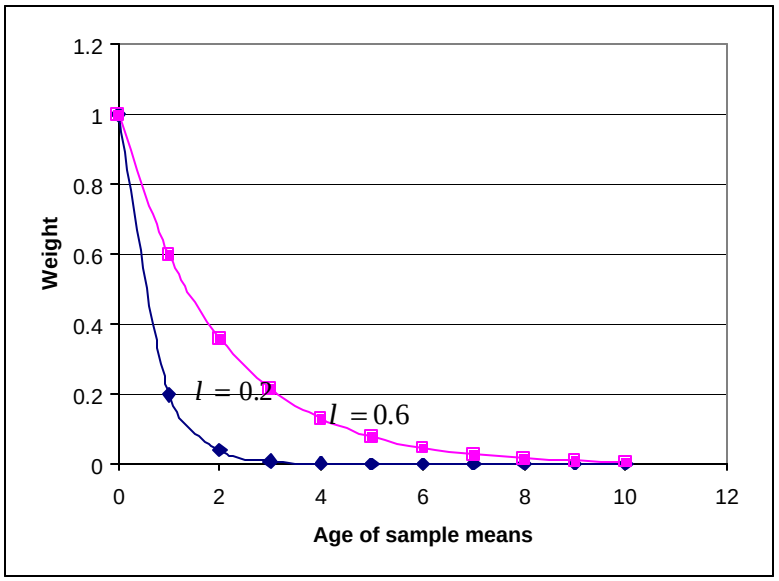


Figure 3.6 Weights of Past Sample Means of EWMA chart

Therefore, the EWMA control chart would be constructed by plotting z_i versus the sample number i (or time). The control limits for the EWMA control chart are as follows.

$$\text{UCL} = m_0 + Ls\sqrt{\frac{l}{(2-l)}[1-(1-l)^{2i}]}, \tag{3.6}$$

$$\text{LCL} = m_0 - Ls\sqrt{\frac{l}{(2-l)}[1-(1-l)^{2i}]}. \tag{3.7}$$

In equation 3.6 and 3.7, the factor L is the width of the control limits. The choice of the parameters L and l will be discussed shortly.

Note that the term $[1-(1-l)^{2i}]$ in equations 3.6 and 3.7 approaches unity as i increases. This means that after the EWMA control chart has been running for several time periods, the control limits will approach the steady-state values given by

$$\text{UCL} = m_0 + Ls\sqrt{\frac{l}{(2-l)}}, \tag{3.8}$$

$$\text{LCL} = m_0 - Ls\sqrt{\frac{l}{(2-l)}}. \tag{3.9}$$

However, Montgomery (2001) recommends using the exact control limits in equations 3.6 and 3.7 for small values of i. This will greatly improve the performance of the control chart in detecting an off-target process immediately after the EWMA is started up. But in this study we used steady state control limits as set forth in equations 3.8 and 3.9.

The EWMA control chart is very effective on detecting small process shifts. The design parameters of the chart are the multiple of the sigma used in the control limits (L) and the value of l. Montgomery (2001) states that it is possible to choose these parameters so that the EWMA control chart yields ARL performance in detecting small shifts that closely approximates CUSUM ARL performance for detecting small shifts.

In general, the values of l in the interval $0.05 \le l \le 0.25$ work well in practice, with l =0.05, l =0.10, and l =0.20 being popular choices. A good rule of thumb is to use smaller values of l to detect smaller shifts.

It is well-known that the Shewhart control chart for individual samples is very sensitive to nonnormality in the sense that the actual in-control ARL (ARL_0) would be considerably less than the expected value based on the assumption of a normal or Gaussian distribution (Montgomery, 2001). But an EWMA chart with small l value is known for its robustness to nonnormality. Borror et al (1999) compare the ARL performance of the Shewhart individual chart and the EWMA control chart in situations of nonnormal distributions. Specifically, they use the gamma distribution to represent the case of skewed distributions and the t distribution to represent symmetric distributions with heavier tails than normal. In their study they draw upon two aspects of the information. First, even moderately nonnormal distributions have the effect of greatly reducing the in-control ARL of the Shewhart chart for individuals. This will, of course, dramatically increase the rate of false alarms. Second, an EWMA with l =0.05 or l =0.10 and an appropriately chosen control limit will perform very well against both normal and nonnormal distributions. Furthermore, the shift detection properties of the EWMA are uniformly superior to the Shewhart chart for individuals.

3.4 BATCH MEAN CHART

Runger and Willemain (1996) proposed a control chart based on unweighted batch means (UBM) for monitoring autocorrelated process data. The batch means approach has been used extensively in the analysis of the output from computer simulation models, which is another area in which highly autocorrelated data often occurs. The UBM chart breaks successive groups of sequential observations into batches, with equal weight assigned to every point in the batch. Let the h^{th} unweighted batch mean be

$$\overline{X}_h = \frac{1}{b}\sum_{j=1}^{b} X_{(h-1)b+j} \quad h = 1, 2, \ldots. \tag{3.10}$$

The important implication of equation (3.10) is that although one has to determine an appropriate batch size b, it is not necessary to construct an autocorrelated model of the data.

Runger and Willemain (1996) show that the batch means can be plotted and analyzed on a standard individuals control chart. Distinct from residuals plots, UBM charts are distinct from residuals plots in retaining the basic simplicity of averaging observations to form a point in a control chart.

Procedures for determining an appropriate batch size have been developed by researchers in the simulation area. These procedures are empirical and do not depend on identifying and estimating a time series model.

Runger and Willemain (1996) provided a detailed analysis of batch sizes for AR(1) models. They recommend that the batch size be selected so as to reduce the lag 1 autocorrelation of the batch means to approximately 0.10. They suggest starting with b=1 and doubling b until the lag 1 autocorrelation of the batch means is sufficiently small. This parallels the logic of the Shewhart chart in that larger batches are more effective for detecting smaller shifts; smaller batches respond more quickly to larger shifts.

3.5 AVERAGE RUN LENGTH AND FALSE ALARM

One of the most important properties associated with any SPC chart is the average run length (ARL). The ARL is the average number of points that must be plotted before a point indicated an out of control condition (Montgomery, 2001). The count of the run

length is initiated at the beginning of a production run or at the first observation after an out-of-control signal has occurred.

When the process is in-control, the average run length should be long but is not infinite because there is a probability (a) of a point being outside the limits even when the process is in control. This is referred to as a **false alarm** and this in-control ARL is called ARL_0.

When the process is out of control, the average run length should be short since this is a correct alarm. This out of control ARL is called ARL_1. ARL_1 is also average run length, but the difference is that there is a signal. In other words, ARL_1 could be called the average **detection time** because the run length in this situation is the time to detect a signal.

Figure 3.7 shows the relation of detection time and run length.

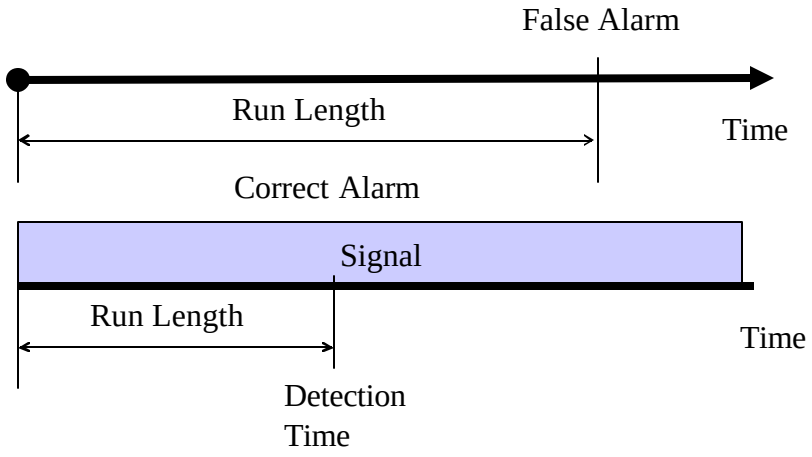


Figure 3.7 Run Length and Detection Time

CHAPTER 4

AN SPC INTRUSION DETECTION APPROACH AND A PRELIMENARY CASE STUDY

This chapter describes the preliminary results of applying SPC charting methods for intrusion detection. In particular, it uses the BSM host audit data from MIT's Lincoln Lab and applies the Shewhart chart, the Cusum chart, and the EWMA chart to detect a DOS intrusion attack. These SPC techniques were applied based on a general SPC intrusion detection approach that is proposed in this thesis. Section 4.1 describes the flow chart of this SPC intrusion detection approach, which consists of three main steps. To illustrate in detail the steps involved in this approach, a case study in the monitoring of a computer information system is introduced. Section 4.2 describes the first step, which is to determine the objective, and illustrates this with an example. Section 4.3 describes the second step, data preparation, and illustrates its execution. Section 4.4 describes the third step, which entails the construction of SPC charts and illustrates this step with an example. Section 4.5, summarizes the results of the case study.

4.1 INTRODUCTION

SPC techniques are very powerful tools for detecting changes in manufacturing processes. This chapter applies these SPC techniques for intrusion detection. The general approach proposed for using SPC techniques in intrusion detection is described in Figure 4.1.

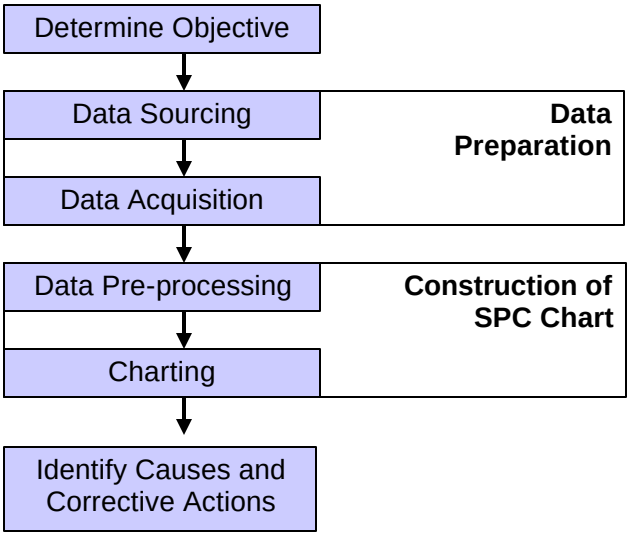


Figure 4.1 A General SPC Intrusion Detection Process

The first step is to *Determine the Objective.* A common objective of an intrusion detection process is to detect the intrusion as fast as possible with the fewest possible false alarms.

The second step is *Data Preparation*. In many data mining methods, including statistical and computational methods, the data preparation step is often difficult and time consuming. The *Data Preparation* step is divided into two steps, *Data Sourcing* and *Data Acquisition*. *Data Sourcing* refers to identifying the sources of data and selecting the target data. *Data Acquisition* refers to transforming the target data into the input data to be used in the SPC methods. This step includes downloading, parsing, and data processing work.

The third step, *Construction of SPC Charts*, is divided into two steps, *Data Pre-processing* and *Charting*. As described in Chapter 3, each SPC procedure includes a

monitoring statistic and a set of control limits for the monitoring statistic. These two elements of the SPC procedure are prepared in the *Data Preprocessing* step. In the *Charting* step, the charting methods with the control limits previously constructed are applied to the monitoring process. Finally, the *Identify Causes and Corrective Actions* step is applied.

In the following discussion, this SPC intrusion detection approach is applied to a typical UNIX computer information system and details each step in the process.

4.2 DETERMINE OBJECTIVE

The objective of the intrusion detection is to detect the intrusion as quickly as possible. This is common to most intrusion detection systems. It is generally true that the faster an intrusion is detected, the better. An intrusion that escapes quick detection can cause many problems.

In the case of a DOS attack, the complete information system can be corrupted by an attack that is not detected quickly. Denial of service is very serious for financial systems, for military command and control systems and for commercial systems. If a DOS attack is detected quickly, the administrator of the information system can take corrective action to prevent or minimize damage from the attack.

Although the goal is to detect intrusions as quickly as possible, the disruptions caused by false alarms cannot be ignored. A false alarm happens when the detection system sets off an alarm for a nonexistent intrusion by the probabilistic nature of these systems; however, false alarms require actions that are time consuming and costly. Consequently,

an effective intrusion detection system will also be a system that minimizes the occurrence of false alarms.

Thus the objective of an SPC intrusion detection approach is to detect the intrusion as fast as possible with minimal occurrence of false alarms. As described in Chapter 3, minimizing false alarms is an alias for maximizing ARL_0 in the SPC area.

4.3 DATA PREPARATION

4.3.1 Data Sources

A typical information system consists of host machines (e.g. machines running a UNIX operation system or machines running the Windows NT operating system) and communication links connecting these host machines to form a computer network. Currently, two sources of data are widely used to capture activities in an information system to permit intrusion detection: network traffic data and audit trail data (audit data). Network traffic data contains data packets traveling over communication links between host machines, and thus capture activities over communication networks. TCPDUMP and *Sendmail* are examples of network traffic data.

Audit trail data captures activities occurring on a host machine. In this study, we use audit data from a UNIX-based host machine (specifically a Sun SPARC 10 workstation with the Solaris operating system), and focus on intrusions into a host machine that leave trails in the audit data. The Solaris operating system from Sun Microsystems Inc. has a security extension called the Basic Security Module (BSM). BSM satisfies the C2 Auditing capabilities of TCSEC (Trusted Computer System Evaluation Criteria) published by NSCC (National Computer Security Center)

The BSM extension supports the monitoring of activities on a host by recording security-relevant events in system call level. There are more than 250 different types of BSM auditable events, depending on the version of the Solaris operating system. A BSM audit record for each event contains a variety of information, including the event type, user ID, group ID, process ID, session ID, the system object assessed, the time when each event occurred, etc. In this research, because event intensity is of interest, only the *time of event*, one of the most critical characteristics of an audit event, is extracted and used. Therefore, activities on a host machine are captured through a continuous stream of audit events, each characterized by the time of the event.

Figure 4.2 shows an example of a typical BSM file. From Figure 4.2, we know that BSM audit data consists of a header field, subject field and return field, and the header field contains information on the event name, event data and event starting time. This header field is used to calculate event intensity.

```
file,Thu 01 Jun 2000 09:59:38 PM EDT, + 391003 msec,
header,111,2,execve(2),,Thu 01 Jun 2000 09:59:41 PM EDT, + 220000000 msec
path,/usr/bin/finger
attribute,100555,root,bin,26738688,74333,0
subject,root,root,other,root,other,648,281,0 0 localhost
return,success,0
header,61,2,exit(2),,Thu 01 Jun 2000 09:59:41 PM EDT, + 240000000 msec
subject,root,root,other,root,other,648,281,0 0 localhost
return,success,0
header,79,2,fork(2),,Thu 01 Jun 2000 09:59:57 PM EDT, + 860000000 msec
argument,0,0x289,child PID
subject,root,root,other,root,other,580,281,0 0 localhost
return,success,0
```

Figure 4.2 An Example of BSM File

4.3.2 Data Acquisition

The large datasets used in this thesis are the Pascal BSM audit datasets, which are obtained from MIT's Lincoln Laboratory. Pascal is the name of the host machine of the network constructed by the MIT Lincoln Laboratory in order to simulate the environment of the network in the real world, and thus provide a test bed of comprehensive evaluations for various intrusion detection systems. The entire datasets consist of seven five-day weeks (Monday through Friday). Two weeks of these datasets were downloaded from MIT Lincoln Laboratory.

After downloading the datasets, MS ACCESS database software and MATLAB programs were used to calculate the event intensity. The event intensity means how many events occurred in a unit of time. A second was used as the time unit in this thesis because the BSM audit data was recorded with this unit of time scale.

The 10 days of data were downloaded and the event intensity raw data files developed for each day. These operations took two months, including downloading time, coding and debugging, and selecting files as the training data samples and testing data samples. The event intensity raw data files were saved in the data repository for future research. The operational procedure was as follows:

- Download BSM file from MIT-LL
- Query the header files with MS ACCESS
- Calculate event intensity
- Save the event intensity raw data in the data repository

The data repository (Appendix 4.A) consists of 10 days of event intensity data files and also of two training samples and two testing samples. After the graphical analysis of the event intensity data file for the 10 days, we concluded there were two periods of *Busy* and *Idle*. The *Busy* period began at 8 a.m. and ended at 4 p.m., which is the start of the idle period that continues until 2 a.m. During the *Busy* period, the average of event intensity is twice as large as that of the *Idle* period.

By using the attack table furnished by MIT's Lincoln Laboratory and 10 days of event intensity files, we selected the pure samples and the attack samples to be used for SPC intrusion detection methods. These were recorded in the data repository (Appendix 4.A). Table 4.1 summarizes the four sample files with two categories. *Pure* means there is no attack within the sample, and *Attack* means there is an attack. *Idle* and *Busy* denotes from which period the sample was drawn. From these samples, we picked idle period samples and used them in the next Charting step. Figure 4.3 shows Pure_sample_data and Attack_sample_data

Table 4.1 Raw Data Samples in Data Repository

Sort	Idle	Busy
Pure	Idle_pure_sample	Busy_pure_sample
Attack	Idle_attack_sample	Busy_attack_sample

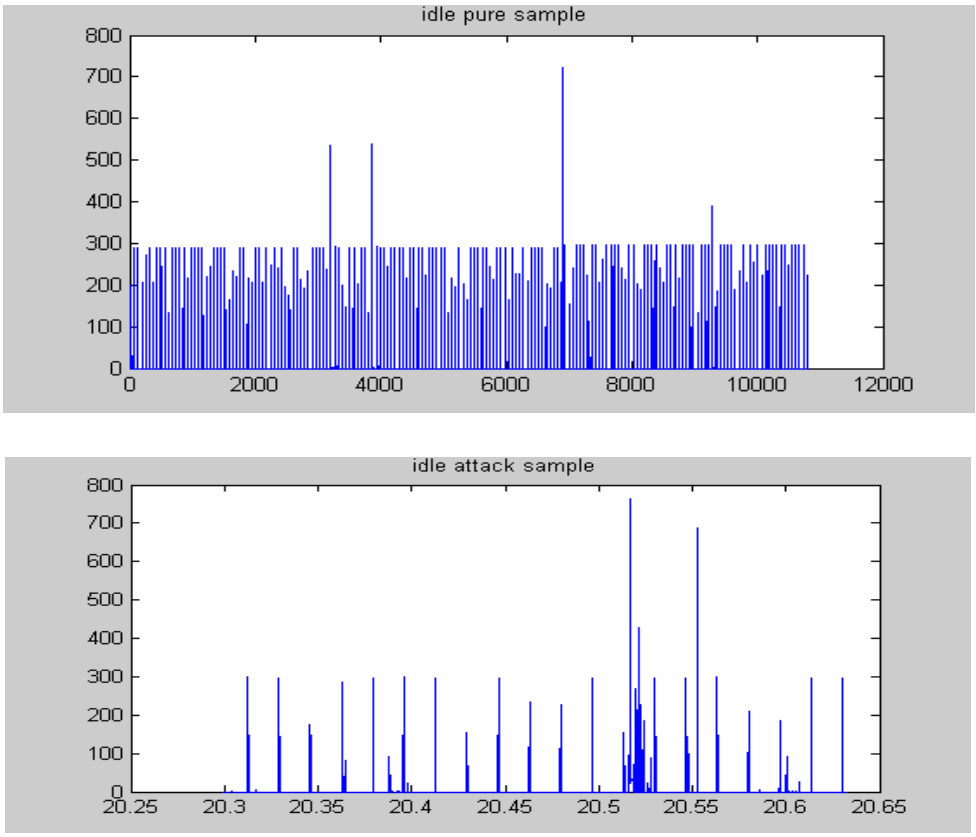


Figure 4.3 Pure_sample_data and Attack_sample_data

4.4 CONSTRUCTION OF SPC CHARTS

To apply SPC methods we need first to set up control limits. Control limits can be constructed by estimating the mean and variance of the monitoring statistic. If the raw data are independent and nicely behaved, we may use the raw data as monitoring statistics with conventional methods. But if the raw data is not independent, then we need to do preprocessing before applying the SPC methods.

Figure 4.4 shows the flow of construction of SPC Chart.

In this section, the SPC charting methods are applied to the sample data that are in Section 4.3. The sample data are the idle samples with attack (Attack_sample_data) or without attack (Pure_sample_data) (see Table 4.1). To construct the control limits, variance is first estimated from Pure_sample_data. With the control limits established, we can calculate the number of false alarms and the detection time, which we then use as performance measures to evaluate the charting methods.

As illustrated below, we will apply three typical charting methods to the data. They

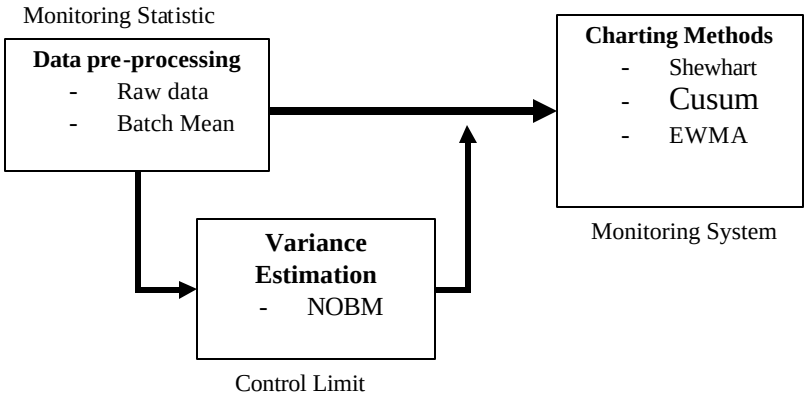


Figure 4.4 Construction of SPC Chart for Intrusion Detection

are the Shewhart chart, the Cusum chart, and the EWMA Chart. Before applying the methods, we will first discuss the data preprocessing step and the reasoning behind it.

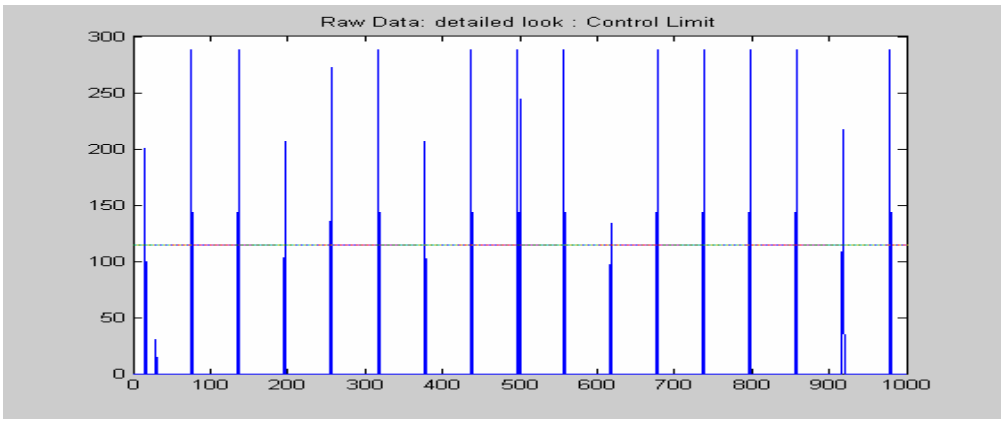
4.4.1 Data Pre-processing

The purpose of data preprocessing is to smooth the data and to reduce correlation. Some SPC methods, such as the Shewhart chart, require data to be independently and identically distributed. The preprocessing step is needed to convert the data to be independent before the Shewhart chart is applied.

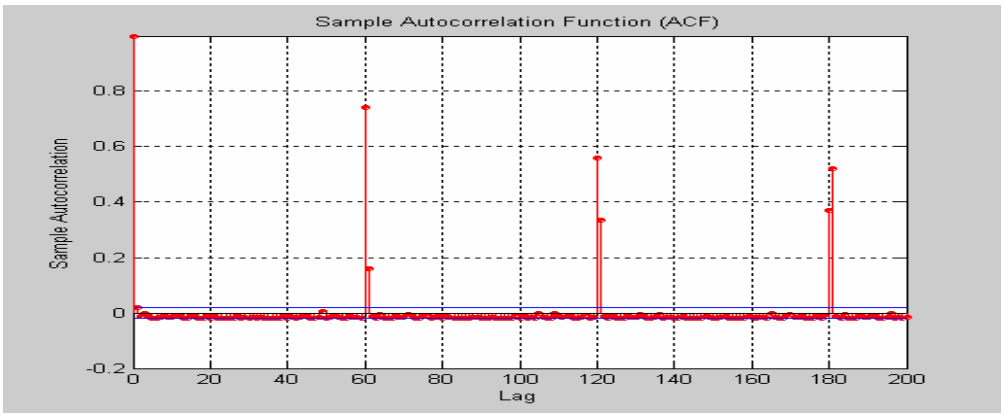
Figure 4.5 shows some raw data plots. Figure 4.5(a) shows the peaky nature of the raw data with a maximum height of about 300. Figure 4.5(b) plots the autocorrelation function of the raw data. It shows that the raw data are autocorrelated with a time lag of 60 and 61 seconds. Because the raw data has autocorrelation and a peaky nature, data preprocessing is needed.

The cause of the peaky nature with a lag of 60 seconds autocorrelation is that the system makes log files every one minutes. This creates about 300 events in the BSM audit file, and this creates peak. The work is done mostly in a second, but sometimes the work done requires two seconds. If this split happens, the two adjacent peaks of event intensity also happen.

Because of the cyclic peaky nature of the raw data, it is difficult to apply traditional control charts directly to the raw data. For example, when the Shewhart or EWMA charts are applied to the raw data, the charts often generate false alarms at every peak. As shown in Figure 4.5 (a), the horizontal line is the control limit of a Shewhart chart and was



(a)



(b)

Figure 4.5 Raw Data: (a) Raw Data without Intrusion, (b) Autocorrelation Function.

obtained based on conventional variance estimation. It was found that every peak was above the control limit. These peaks were considered false alarms because the data were collected without intrusion.

An investigation was undertaken to learn why the peaky data always causes false alarms when we a conventional control chart is used. To explain this symptom, we created a potential data named constant peak cycled data (CPCD). CPCD has a constant peak height C and a cycle length L. Then CPCD are autocorrelated with cycle length L,

and the standard deviation of the CPCD is $C\sqrt{\frac{L-1}{L^2}}$. This means that when we use a 3-s control limit, if the cycle length L is 60, the standard deviation is very small, i.e.,

$$\hat{s} = C\sqrt{\frac{60-1}{60^2}} = 0.128 \cdot C,$$

thus, the 3-s control limit is

$$\hat{m} + 3\hat{s} \approx \frac{C}{60} + 3(0.128 \cdot C) = 0.40 \cdot C.$$

Therefore, the control limit is smaller than the peak value ($0.40 \cdot C <$ C). The details about CPCD are described in Appendix 4.B

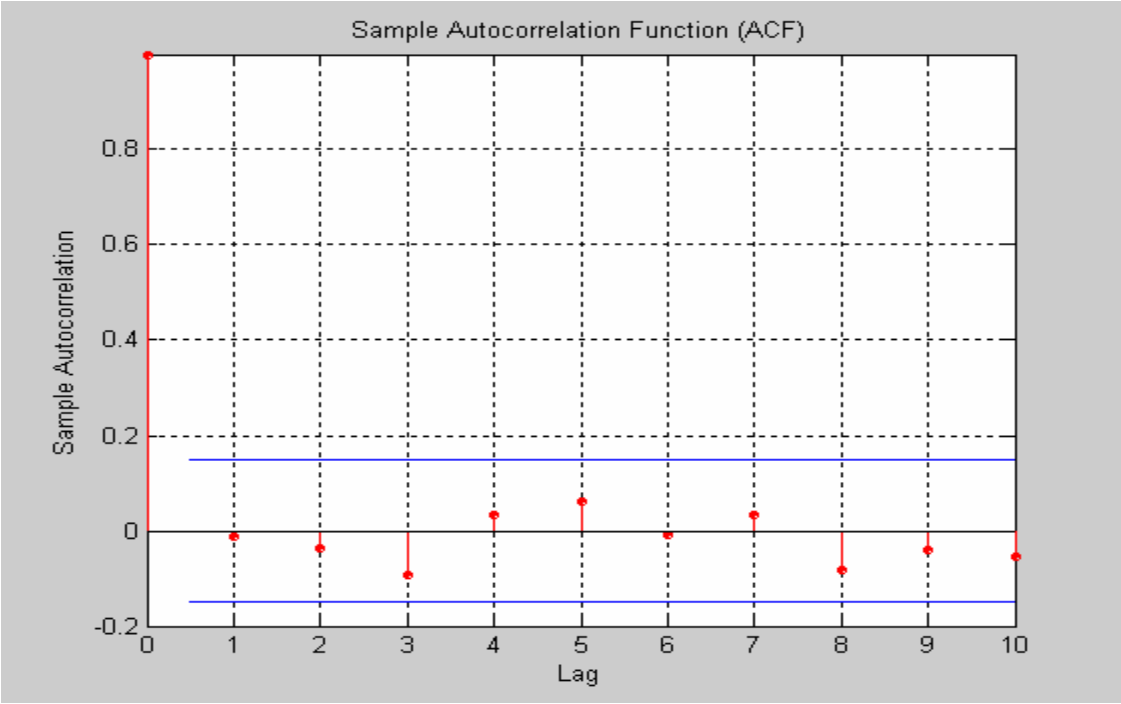


Figure 4.6 Sample Autocorrelation Function for Batch Size 60 Preprocessed Data of the Pure_sample

For this reason, the use of conventional SPC methods requires preprocessing the raw data. To preprocess the raw data, we considered nonoverlapping batch mean (NBM) to smooth the raw data and remove the autocorrelation. Because the raw data have a peaky nature with a 60 cycle, we picked non-overlapping batch sizes as multiples of 60 seconds like 60, 120, 180, 240, and 300 seconds. After experimenting with different batch sizes with the real data, it was concluded that 60 and 120 seconds were the best lengths because they could eliminate autocorrelation effectively and are not so large as to be costly. Figure 4.6 shows the autocorrelation function values for NBM (60) of Pure sample. This shows that the batch size with 60 seconds gives uncorrelated data.

As mentioned earlier, we needed first to estimate the variance before we computed the control limits. We applied the nonoverlapping batch mean (NOBM) variance estimation method to the raw data to find the asymptotic variance and found that the smallest batch size that provides independency is 60 and the asymptotic variance, $\hat{s}^2_{x,\infty} =$ 238.02.

Variance estimation using the NBM (60) preprocessed data gave us the asymptotic value of $\hat{s}^2_{y,\infty} = 238.02$ and the smallest batch size = 1 that provides independency. This result showed us that the 60-second batch is enough to resolve the peaky nature and gives independence. Thus we can use the preprocessed NBM (60) as a monitoring statistic. The variance estimate of NBM (60) $= 1.9917^2$, and the sample mean is 5.0552.

4.4.2 Charting

After data preprocessing, traditional SPC methods can be applied to monitor the data. Three typical SPC methods were considered in our example, the Shewhart chart, EWMA chart, and Cusum Chart.

4.4.2.1 Shewhart Chart

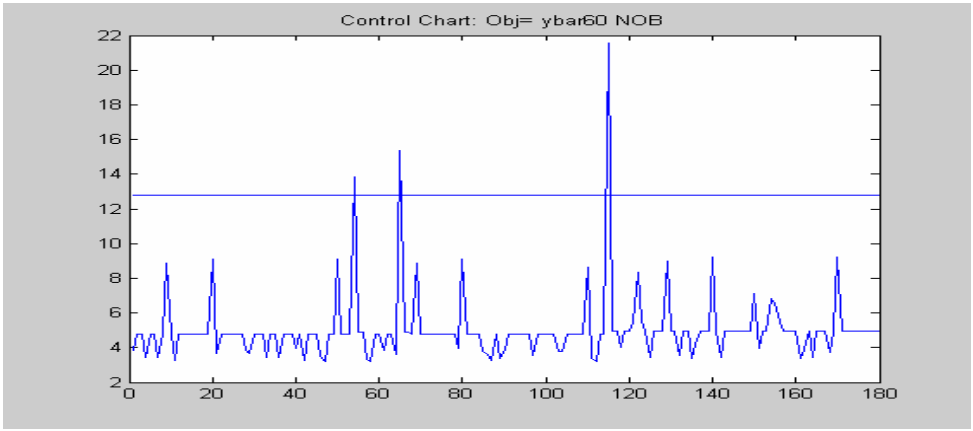
We applied the Shewhart chart to the preprocessed data with 3-sigma control limits. It is interesting to note that the Shewhart chart on the NBM preprocessed data is equivalent to the batch mean chart proposed by Runger & Willemain (1995).

The sample data used here were chosen from the typical samples of the prepared data described in Section 4.2.2 and summarized in Table 4.1. The chosen samples were the idle data without attack (Pure_sample_data) and idle data with attack (Attack_sample_data). Several samples had been investigated, and the results were similar. Only the results of one sample from the pure sample and one attack sample will be reported below

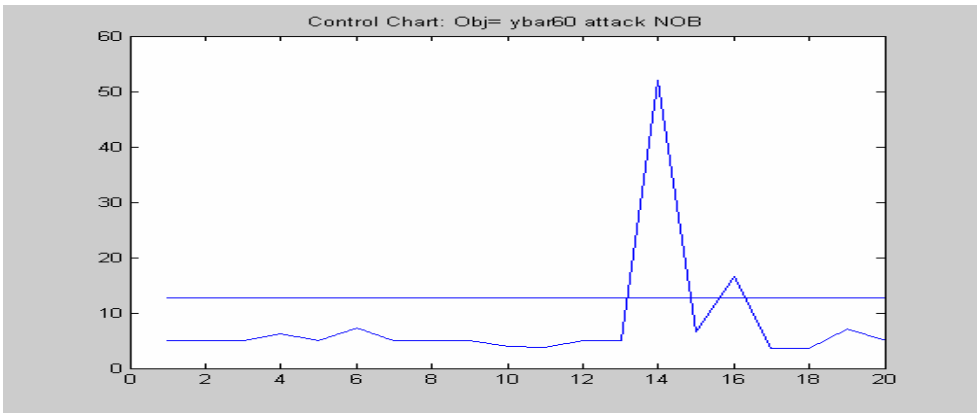
Shewhart Chart on Batch Mean (60)

Figure 4.7 shows the result of the Shewhart chart applied to the preprocessed data. The monitoring statistic is the nonoverlapping batch mean (NBM) with a batch size of 60 seconds. The sample standard deviation of the statistic is 1.9917. The control limit was calculated based on Pure_sample_data.

For the pure sample, there were 3 false alarms, and for the attack sample, the attack was detected at the 14^{th} batch. The batch size was one minute (= 60 seconds). So the detection time was 14 minutes.



(a)

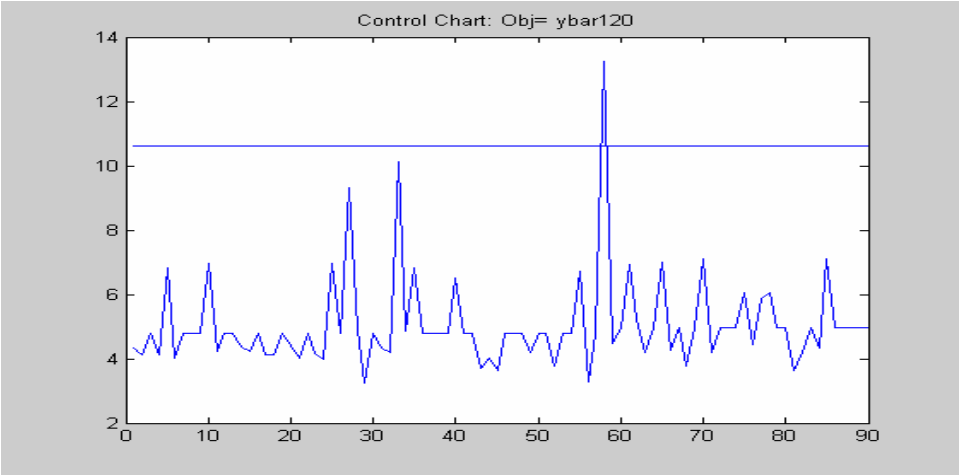


(b)

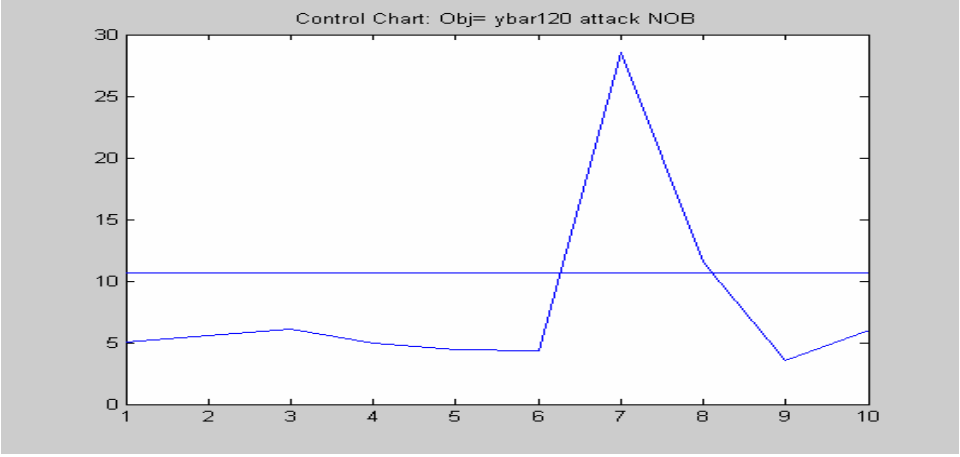
Figure 4.7 Shewhart chart for NBM (60): (a) Pure_sample, (b) Attack_sample

Shewhart Chart on Batch Mean (120)

Figure 4.8 shows the result of the Shewhart chart applied to preprocessed data with a batch size of 120 seconds. The monitoring statistic was the batch mean of size 120 seconds (2 minutes). The standard deviation of the monitoring statistic was calculated to be 1.4323.



(a)



(b)

Figure 4.8 Shewhart chart for NBM (120): (a) Pure_sample, (b) Attack_sample

For the pure sample, there was one false alarm, and for the attack sample, the attack was detected at the 7^{th} batch. The batch size was two minutes (= 120 seconds), so the detection time was 14 minutes.

The results of the Shewhart chart are summarized in Table 4.2. For both charts, the detection times were the same, but the number of false alarms differs. The number of false alarms with NBM (120) is lower than with NBM (60). It may be because the NBM (120) was smoother than NBM (60) as a resulting of having a bigger batch size.

It was interesting that the detection time was 14 minutes. This is because the attack was detected at the end of the batch and the ends occurred at 14 minutes. In other words, if the batch scale is 1 minute, then we can detect the attack with this scale. This is one topic we will study in this thesis.

Table 4.2 Result of Shewhart Charts on NBM (60, 120)

Monitoring Statistic	Charting Method	Pure sample	Attack sample
NBM (60)	Shewhart	3 false alarms	Detects at the14^{th} batch = 14 min.
NBM (120)	Shewhart	1 false alarm	Detects at the7^{th} batch= 14 min.

4.4.2.2 Cusum Chart

In this subsection, we apply the tabular Cusum chart for IID sample with the formula and notation described in Montgomery (2001). The formulation of Cusum is as follows:

$$Cusum_i = \max[\,0, (X_i - m_0 - K) + Cusum_{i-1}\,],$$

where K (=kσ) and H (=hσ) are charting parameters. The choice of K and H values can be found in Montgomery (2001). In our example, we chose two values of k (0.5 and 0.0) and the companion of h value.

The Figure 4.9 shows the result of applying the Cusum chart to the preprocessed data with a batch size of 60 seconds. The monitoring statistic was the batch mean of size 60 seconds (NBM (60)).

For the pure sample, there was one false alarm (starts at 115 and ends at 123) and for the attack sample, the attack was detected at the14th batch. The batch size was one minute (= 60 seconds), so the detection time was 14 minutes.

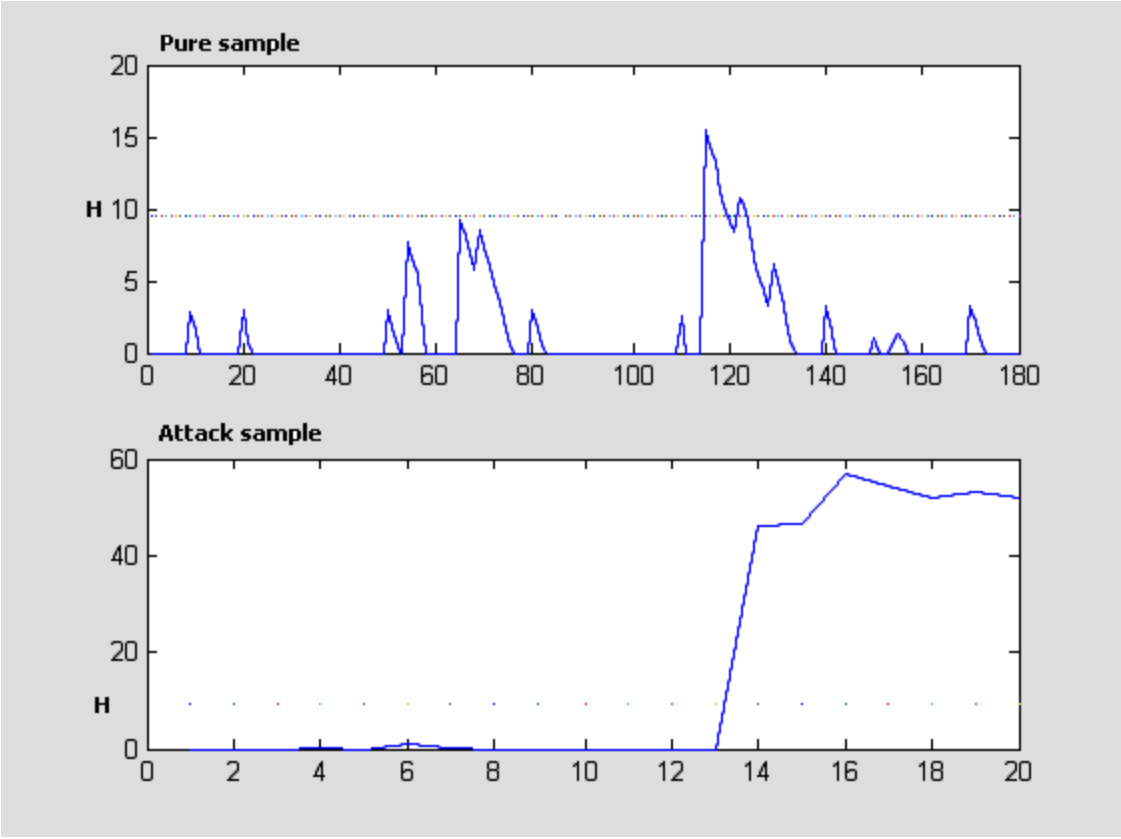


Figure 4.9 Cusum Chart (k=0.5) for NBM (60)

Figure 4.10 shows the result of applying the Cusum chart to the preprocessed data with a batch size of 120 seconds. The monitoring statistic was the batch mean of size 120 seconds (NBM (120)).

For the pure sample, there was one false alarm (starts at 58 and ends at 58) and for the attack sample, the attack was detected at the 7th batch. The batch size was two minutes (= 120 seconds), so the detection time was 14 minutes.

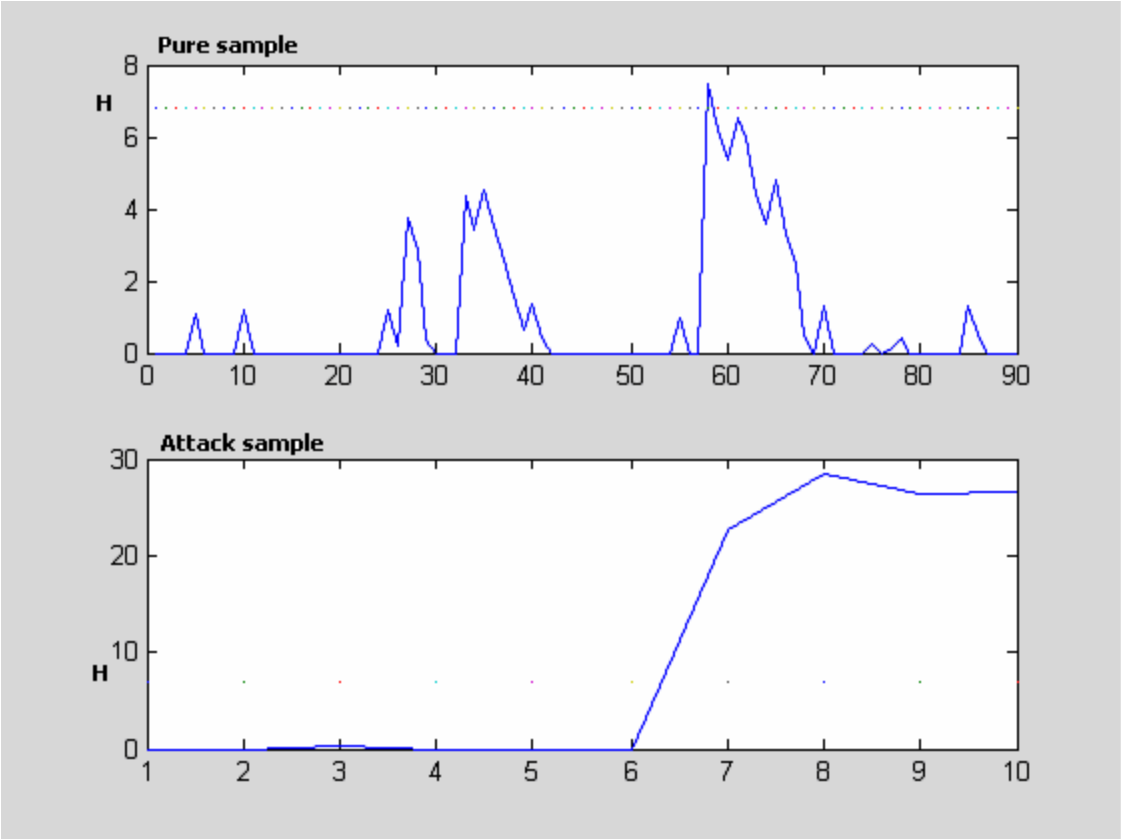


Figure 4.10 Cusum Chart (k=0.5) for NBM(120)

In Figure 4.11 and Figure 4.12, the Cusum method was applied to the two batch means with K=0.0. Since the control limit (H value) was not given in Montgomery

(2001) for K=0, the method developed in Johnson and Bagshaw (1974) was used to determine the control limit.

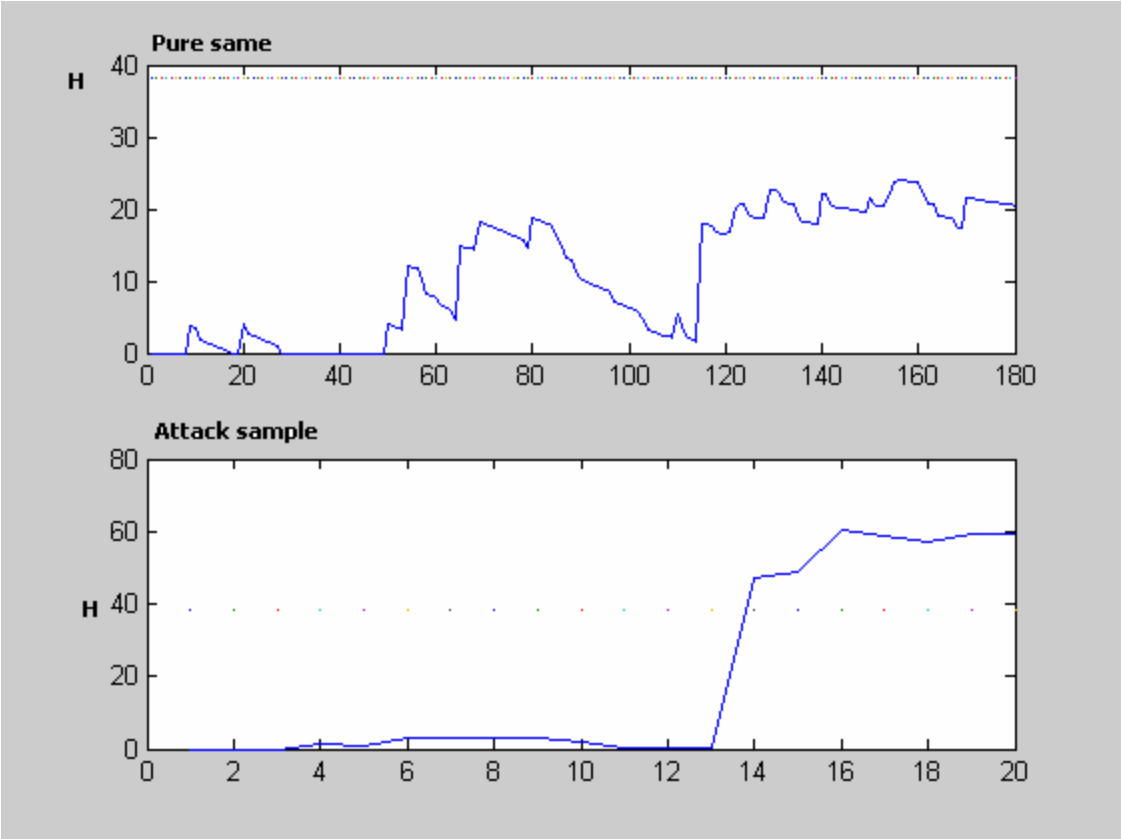


Figure 4.11 Cusum Chart (k=0) for NBM (60)

Figure 4.11 shows the result of applying the Cusum chart (with k=0) to the preprocessed data with a batch size of 60 seconds. The monitoring statistic was the batch mean of size 60 seconds (1 minute).

For the pure sample, there was no false alarm and for the attack sample, the attack was detected at the 14^{th} batch. The batch size was one minute (= 60 seconds), so the detection time was 14 minutes.

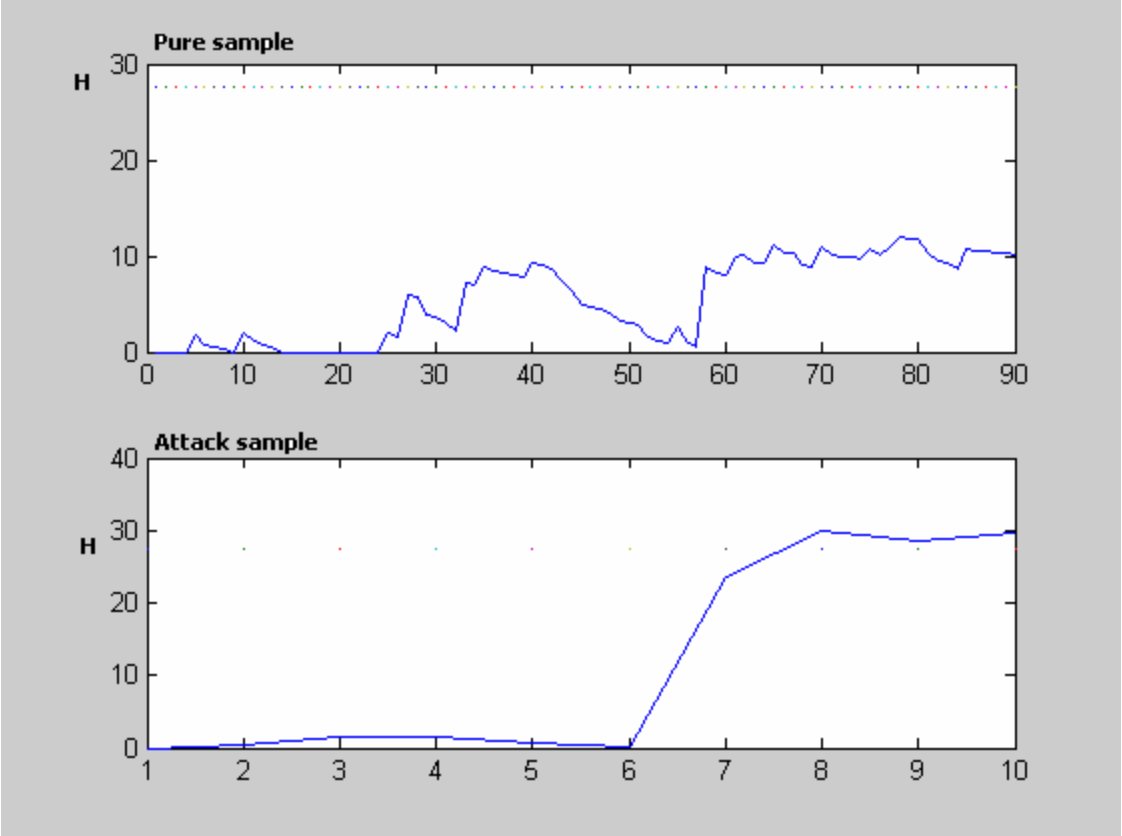


Figure 4.12 Cusum Chart (k=0) for NBM(120)

Figure 4.12 shows the result of applying the Cusum chart (with k=0.0) to the preprocessed data with a batch size of 120 seconds. The monitoring statistic was the batch mean of size 120 seconds (2 minutes).

For the pure sample, there was no false alarm, and for the attack sample, the attack was detected at the 8^{th} batch. The batch size was two minutes (= 60 seconds), so the detection time was 16 minutes.

4.4.2.3 EWMA Chart

In this subsection, the EWMA Chart for IID data was applied to the two preprocessed batch means (NBM (60), NBM (120)).

We applied the EWMA chart to the data based on the formula given in Chapter 3. Here we used the initial value (z_0) as the sample mean of the pure sample and the standard deviation (σ) as the sample standard deviation of the preprocessed data. We chose l = 0.2 and L = 2.86, which gave ARL_0 = 370 as recommended in Montgomery (2001). The formulation of the EWMA monitoring statistics was given as follows

$$EWMA_i = 0.2 \cdot X_i + (1 - 0.2) \cdot EWMA_{i-1},$$

where X_i is the preprocessed batch mean.

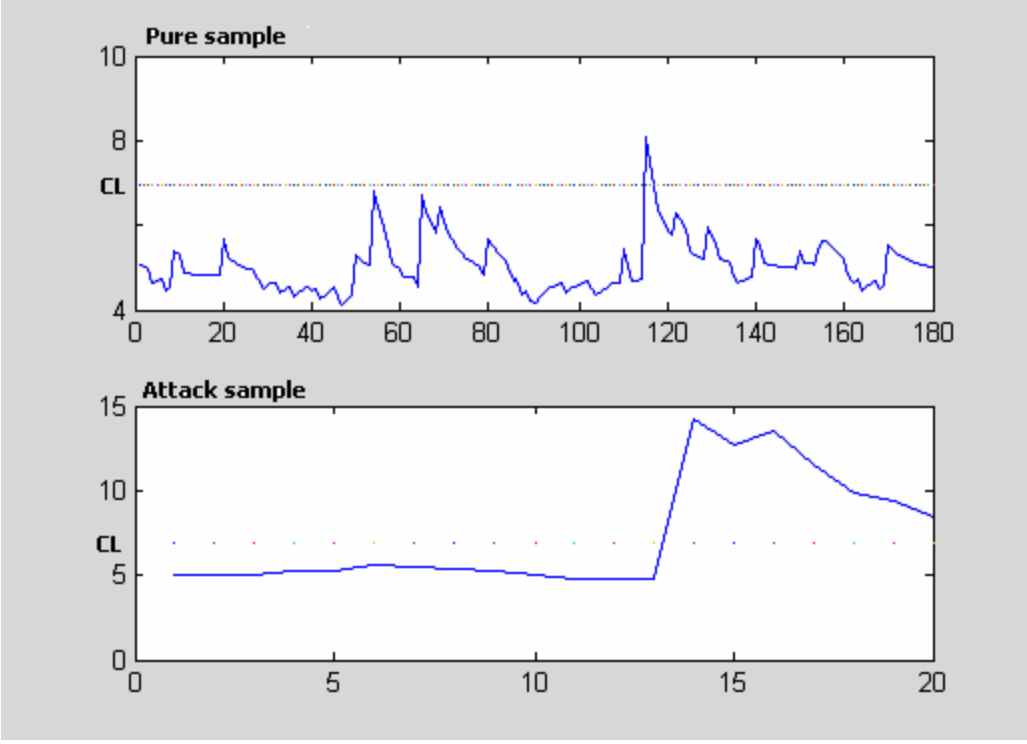


Figure 4.13 EWMA Chart for NBM (60)

Figure 4.13 shows the result of applying the EWMA chart to the preprocessed data with a batch size of 60 seconds. The monitoring statistic was the batch mean of size 60 seconds (1 minute).

For the pure sample, there was one false alarm, and for the attack sample, the attack was detected at time 14th batch. The batch size was one minute (= 60 seconds), so the detection time was 14 minutes.

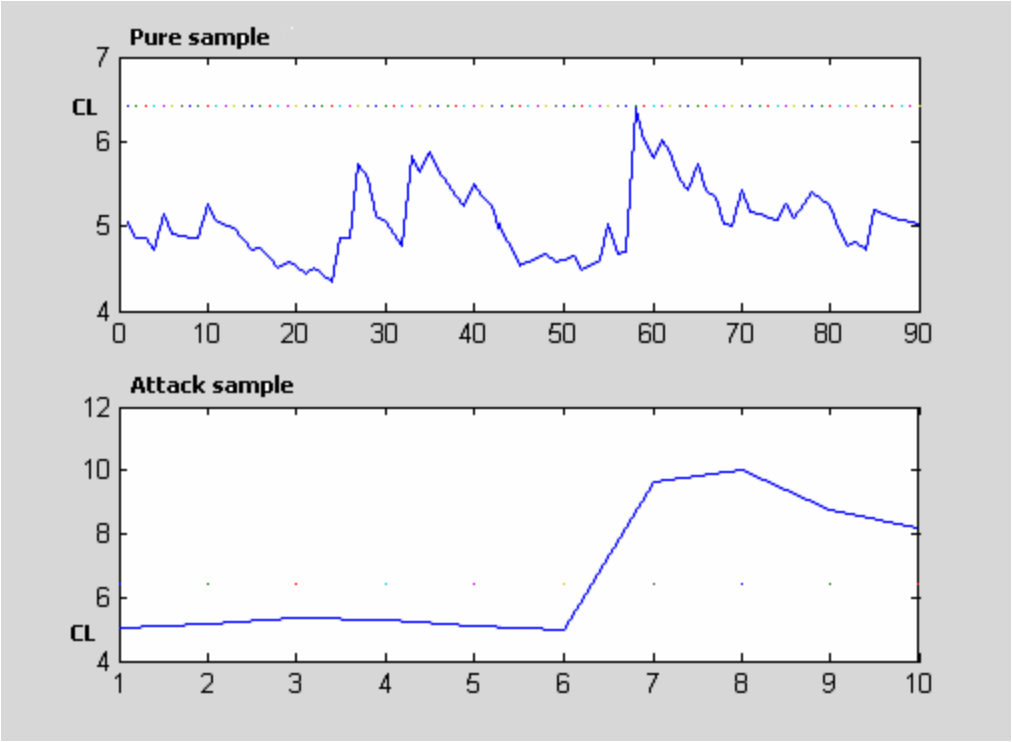


Figure 4.14 EWMA Chart for NBM (120)

Figure 4.14 shows the result of applying the EWMA chart to the preprocessed data with a batch size of 120 seconds. The monitoring statistic was the batch mean of size 120 seconds (2 minutes).

For the pure sample, there was no false alarm, and for the attack sample, the attack was detected at time 7^{th} batch. The batch size was two minutes (= 120 seconds), so the detection time was 14 minutes.

4.5. SUMMARY OF CASE STUDY

We applied three charting methods in the case study. The sample data used in this case study was from the data repository and had been preprocessed so that the monitoring statistics were smooth and approximately uncorrelated.

Table 4.2 shows the summarized results of the case study. The monitoring statistics were batch mean with size = 60 (NBM (60)) and size = 120 (NBM (120)). The charting methods were the Batch Mean Shewhart Chart, the Batch Mean Cusum Chart and the Batch Mean EWMA chart.

Table 4.2 Summary of charting methods

Charting Method	Monitoring Statistic	Number of False Alarm in Pure sample	Detection Time in Attack sample
Shewhart	NBM (60)	3	14 min.
	NBM (120)	1	14 min.
Cusum: k=0.5	NBM (60)	1	14 min.
	NBM (120)	1	14 min.
Cusum: k=0.0	NBM (60)	0	14 min.
	NBM (120)	0	16 min.
EWMA	NBM (60)	1	14 min.
	NBM (120)	0	14 min.

In general, the number of false alarms decreased as batch size increased. This is because of the smoothing effect of batch size. This indicates that the larger the batch size is, the fewer false alarms occur. Overall, the Cusum and EWMA charts had fewer false alarms than the Shewhart chart.

In terms of detection time, all the charts resulted into the same time of 14 minutes, except the Cusum Chart (k=0.0) of NBM (120) data. This happened because the batch mean charting methods only detected the shift at the end of the batch. Because the attack occurred between the 13^{th} and 14^{th} minutes (at the 784^{th} second = 13.06 minutes), and the attack signal was very large, most of the charts detected the signal at the end of the first batch. But the Cusum Chart (k=0.0) of NBM (120) detected the attack at 16 minutes. This may be because the control limit H is too large to detect the attack at the first batch. So the attack was detected at the 16^{th} batch or 960^{th} second.

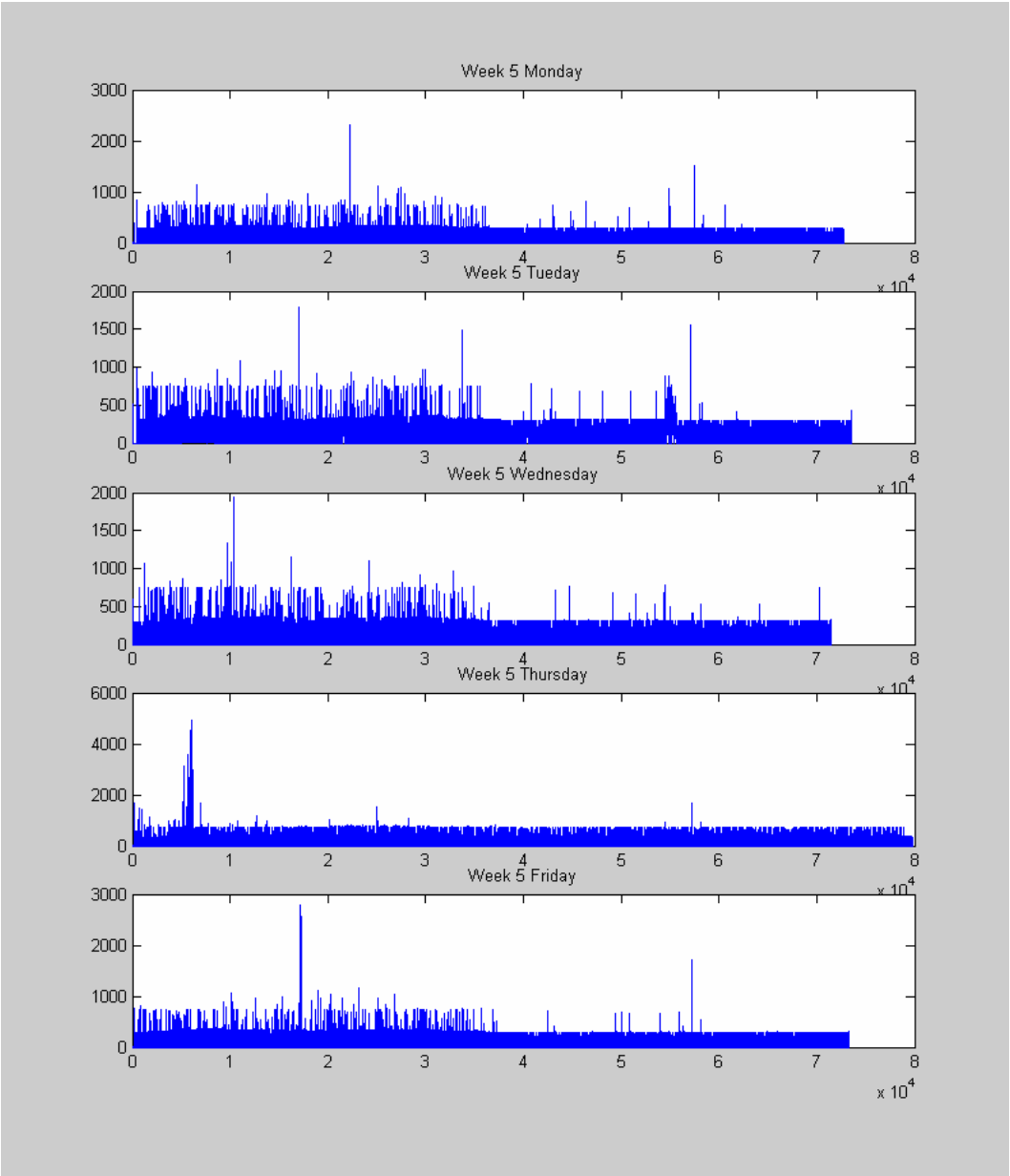
For the monitoring statistic of NBM (60), while the attack occurred at the beginning of the batch, i.e. the 4^{th} second of the 60-second batch, it was not detected until at the end of the batch, i.e. the 60^{th} second of the batch. For the monitoring statistic of NBM (120), the attack occurred in the middle of the batch, i.e., the 64^{th} second of the 120 seconds batch, but it was not detected until at the end of the batch, i.e., the 120^{th} second of the batch.

As expected, as batch size increases, the opportunity that the attack will be detected early decreases. For example, the Cusum Chart (k=0.0) of NBM (120) detected the attack at the end of the second batch. This implies that the smaller the batch size is, the better the detection time performance is.

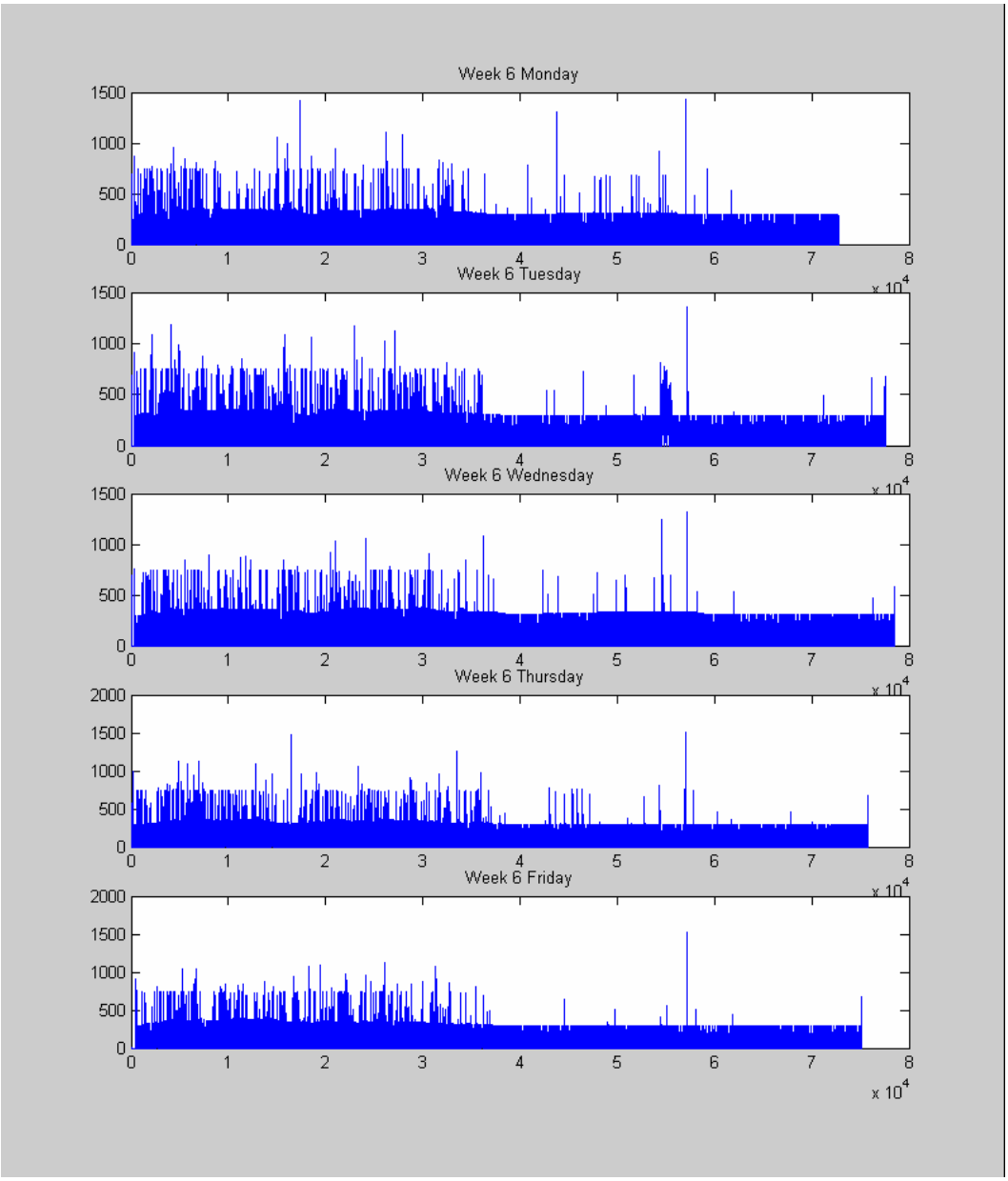
As these two performance measures (the number of false alarms and detection time) conflict with each other, it is often difficult to determine the proper batch size in practice. In the next chapter, we propose a modified batch mean method that rectifies this problem and allows the use of a large batch size without any loss in speed of detection.

APPENDIX 4A. GRAPHS OF EVENT INTENSITY DATA

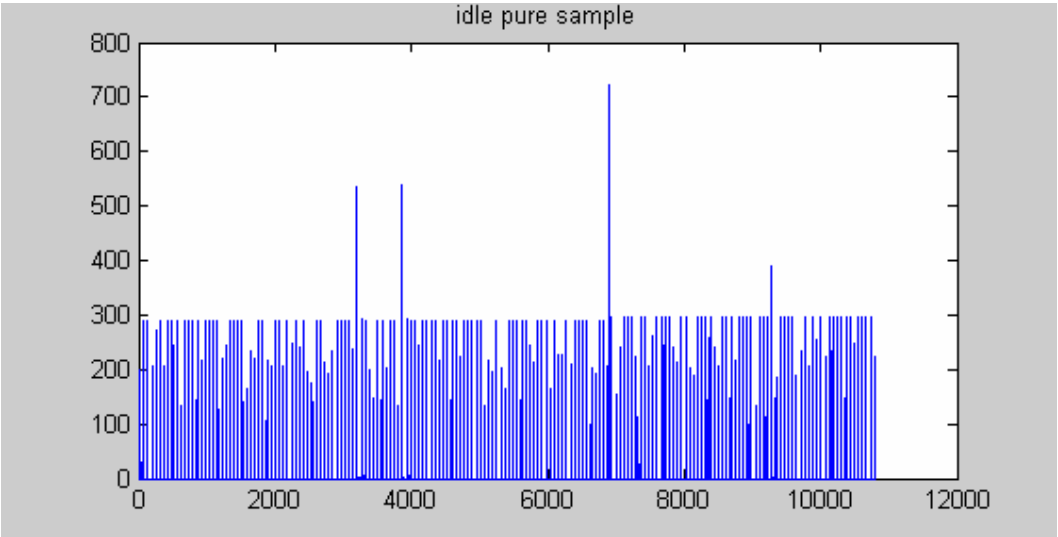
Graphs for 10 days



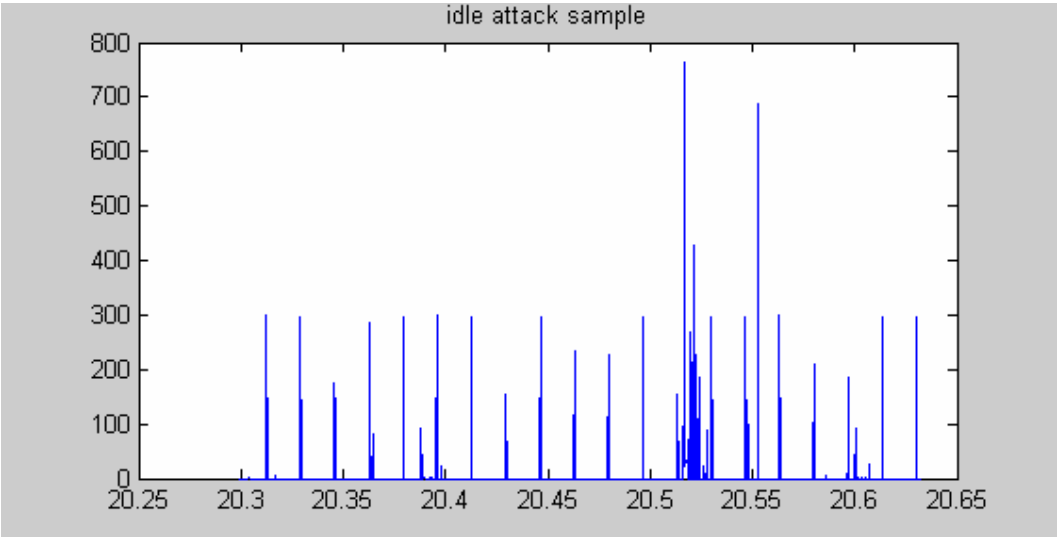
Graphs for 10 days (continued)



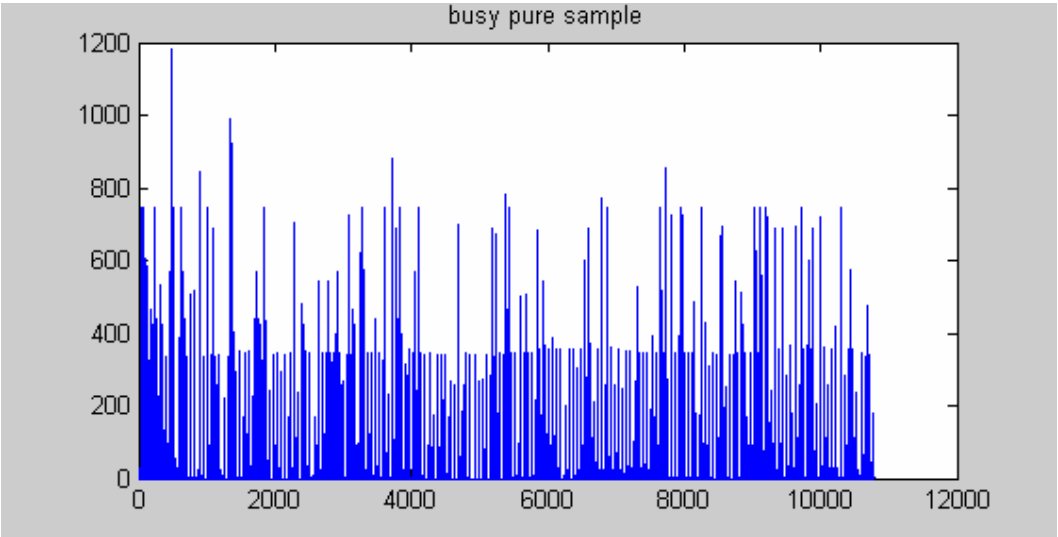
Idle Pure Sample Data



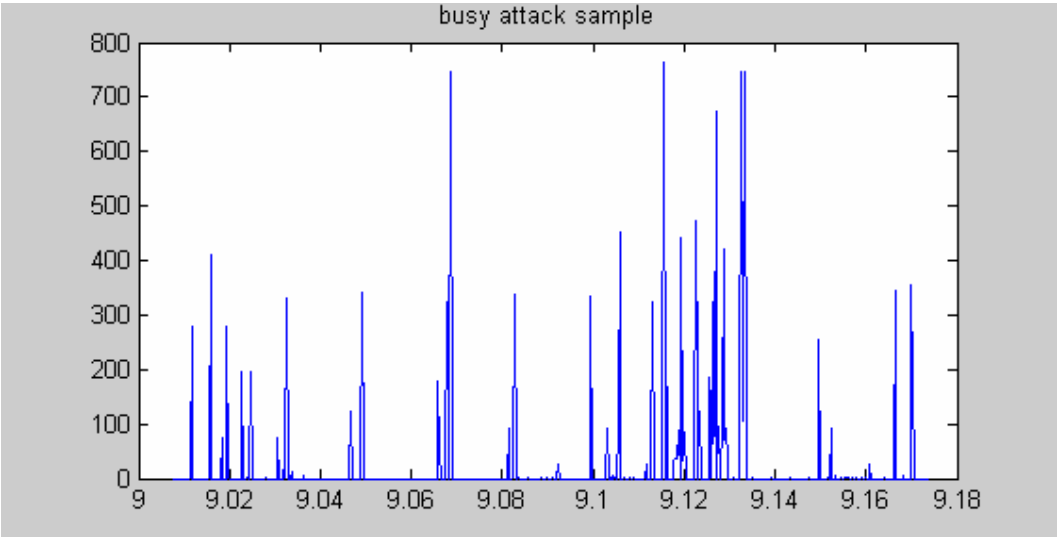
Idle Attack Sample Data



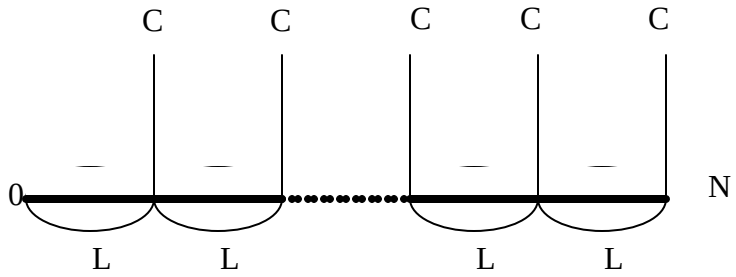
Busy Pure Sample Data



Busy Attack Sample Data



APPENDIX 4B. CPCD (CONSTANT PEAK CYCLED DATA)



CPCD is the potential form of the raw data. CPCD has constant peak value of C and Cycle length of L.

Then the CPCD had the mean = C/L, and the standard deviation $= C\sqrt{\frac{L-1}{L^2}}$.

And lag L autocorrelation is $\frac{N-L}{N}$.

Proof.

(i) **Standard deviation (STD) of CPCD**

$$STD^2 = EX^2 - (EX)^2 = C^2/L - C^2/L^2 = C^2(L-1/L^2),$$

where X are CPCD.

Thus,

$$STD = \sqrt{C^2(L-1/L^2)} = C\sqrt{\frac{L-1}{L^2}}.$$

(ii) **Autocorrelation of CPCD**

Notation:

L: Cycle Time length, or Lag.
C: Constant. The intensity of idle peaks.
N: Number of sequence. Assumed as the multiple of L.

$$r_k = \frac{\Sigma_{i=1}^{N}(Y_i - \bar{Y})(Y_{i+k} - \bar{Y})}{\Sigma_{i=1}^{N}(Y_i - \bar{Y})^2}$$

if k=L,

$$\begin{aligned} r_L &= \frac{\Sigma_{i=1}^{N}(Y_i - \bar{Y})(Y_{i+L} - \bar{Y})}{\Sigma_{i=1}^{N}(Y_i - \bar{Y})^2} \\ \bar{Y} &= \frac{C}{L} \\ (Y_i - \bar{Y})(Y_{i+k} - \bar{Y}) &= Y_i Y_{i+L} - \bar{Y}(Y_i + Y_{i+k}) + \bar{Y}^2 \end{aligned}$$

Nominator

$$\begin{aligned} \Sigma_{i=1}^{N}(Y_i - \bar{Y})(Y_{i+L} - \bar{Y}) &= (\frac{N}{L} - 1)C^2 - \frac{C}{L}(2C(\frac{N}{L} - 1)) + (N - L)(\frac{C}{N})^2 \\ &= \frac{C^2}{L^2}(N - L)(L - 1) \end{aligned}$$

Denominator

$$\Sigma_{i=1}^{N}(Y_i - \bar{Y})^2 = \Sigma Y_i^2 - n\bar{Y}^2 = \frac{n}{L}C^2 - n\frac{C^2}{L^2} = \frac{NC^2(L-1)}{L^2}$$

Thus,

$$r_L = \frac{(N - L)(L - 1)}{N(L - 1)} = \frac{N - L}{N}$$

CHAPTER 5

SIMULATION MODELING FOR SPC INTRUSION DETECTION METHODS

In this chapter, simulation models for SPC intrusion detection methods will be built to compare the performance of various SPC methods. We will develop simulation models based on real data from the BSM file that were recorded by Solaris OS of MIT Lincoln Lab

In Section 5.1, we will describe simulation input modeling for rendering simulation traffic data, and in Section 5.2 we will discuss how to generate traffic data for simulation studies in Section 5.2. Section 5.3 summarizes scenarios that will be tested in Chapter 6.

5.1 SIMULATION INPUT MODELING

For the purpose of rendering realistic simulation traffic data, we first analyze the Pure_sample_data obtained in Chapter 4. The Pure_sample_data are traffic data that contain cycle and noise data but no attack data.

In Chapter 4, we found that the sample data have 60-second cyclic peaks and they are sometimes split in two with an interval of one second. We define cycle data as the cyclic peaks of the traffic data and define noise data as traffic data that are not related with cyclic peaks when there is no attack. From this information, we can discriminate noise data from cycle data for any data without an attack.

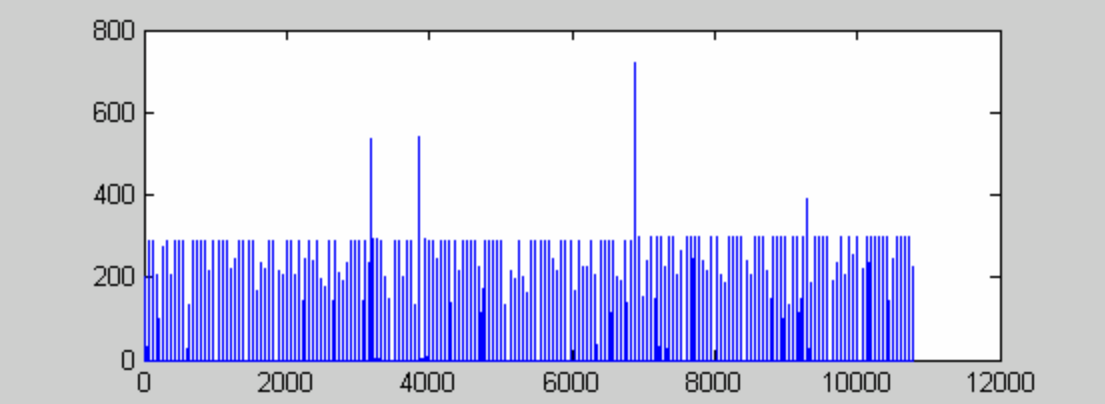


Figure 5.1 Pure_sample_data

For the purpose of constructing input models for cycle and noise distribution, we plot a number of sample data without attack. Figure 5.1 shows the plot of Pure_sample_data from Chapter 4. Other sample data without attacks showed similar pattern.

After investigating the Pure_sample_data second by second, we found that there are 179 cycle peaks and 57 non-zero noise distribution points. For more details, we separated cycle data from noise distribution (Noise_data) in the Pure_sample_data. The graphs of each data type are shown in Figure 5.2.

Although the exact sources of noise are not known, it seems that the noise distribution are from the computer networks because workstations are connected online with other computers through the networks.

In the next two subsections, we will discuss the properties of each data type and fit probability models to the data.

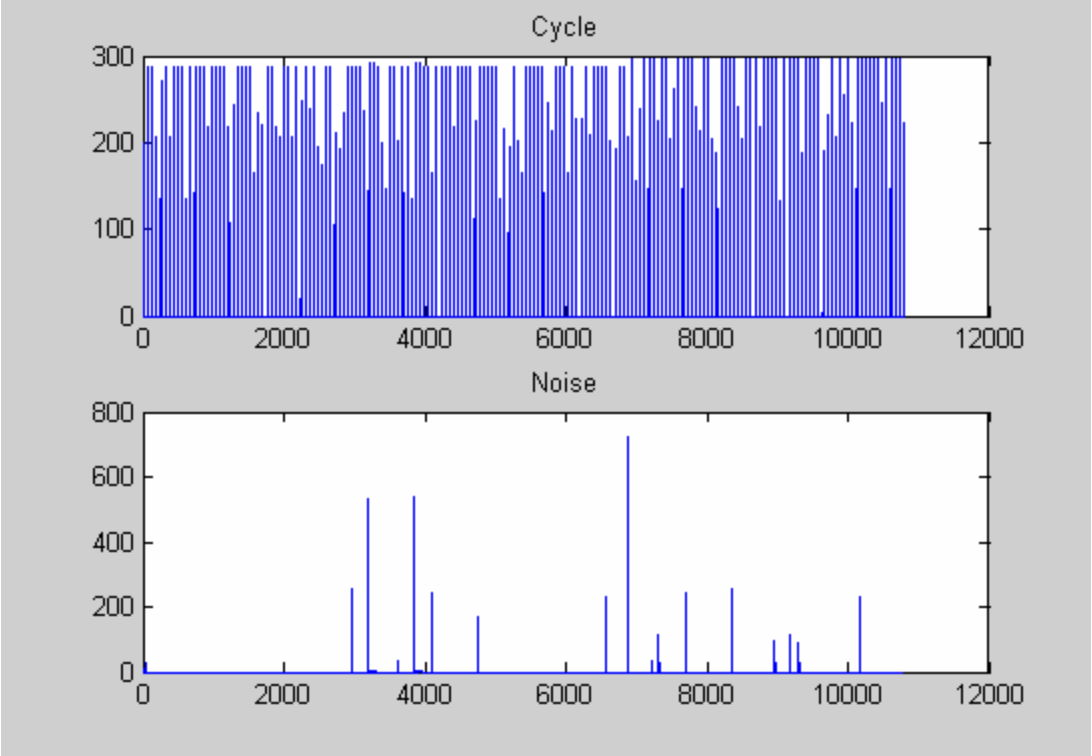


Figure 5.2 Cycle and Noise Data Separated from the Pure_sample_data

5.1.1 Cycle Data

The data we obtained from the BSM file that was recorded by the Solaris OS of MIT Lincoln Lab have cycle peaks, in our case, a 60-second constant cycle time. The Solaris OS performed a specific routine for creating a log file every 60 seconds regardless of the status of the system. We conjecture that this is why there is a 60-second cycle.

Sometimes these cycle data were split into two with an interval of one second. We observed this by analyzing the height of cycle data. Routines performed by Solaris OS usually create around 300 events, although the exact values are 288 or 297 and vary a little bit. However, there were some cases in which the heights were much shorter than 300 at the time when a cycle was expected. For those cases, when we added up the number of events at the cycle period and the next second, the sum of events at those two

consecutive seconds came close to 300 (288 or 297). This strongly implies that a split is possible. Figure 5.3 illustrates this graphically.

In Figure 5.3, the first cycle occurs at 4 seconds and the second cycle occurs at 64 seconds as expected. The heights of both cycles are 288. The third cycle is expected at 124 seconds, but the height of the cycle at 124 seconds is 217, which is much smaller than 288. However, the sum of events at 124 and 125 seconds is 288. In the simulation study, we assumed that the height (events) of each cycle is simply a constant 300.

The split probability (p_s) can be found by counting how many splits occur out of the total number of cycles. For the Pure_sample_data in Figure 5.2, we found that there are 179 cycle peaks and 34 splits, which gives the split probability of 34/179 = 0.20 approximately. In our simulation study, we tested various split probabilities, including this value.

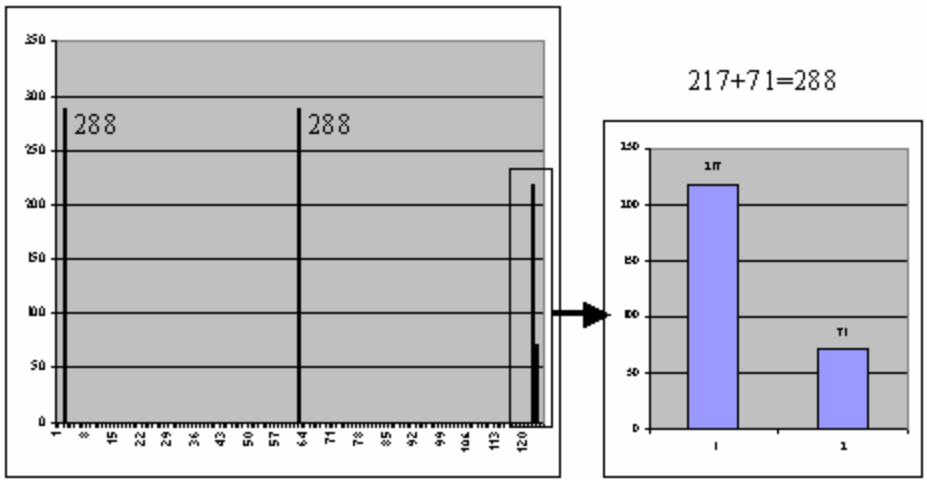


Figure 5.3 Split of Cycle Peak

5.1.2 Noise

As shown in Figure 5.2, noise data points are rare and have a very large variance. To find a good approximated probability model for real noise distribution, we used software packages available for data fitting in simulation area.

In general, a Poisson distribution is a popular choice for network traffic data. However, the variance of our noise distribution is too big to be represented by a Poisson distribution. For example, Noise_data in Figure 5.2 have a mean of 81 and a standard deviation of 154, which is almost twice as large as the mean. A Poisson distribution with a mean of 81 only gives a standard deviation of 9, which is too small compared to 154. Therefore, a simulation model with Poisson distributed noise distribution will easily be misleading, and we should find an appropriate distribution model that is a good representation of our noise distribution.

S-plus and **BestFit** were applied to the noise distribution in Figure 5.2 to find a good fit, but none of the simple standard probability models fitted will with our data. Therefore, we used more complicated input models whose shapes are extremely flexible, such as the Johnson translation system and the Bezier distribution.

The Johnson translation system (Johnson, 1949) is defined as

$$F(x) = \Phi\{g + dg[(s-x)/l]\}, \; -\infty < x < \infty,$$

where Φ is the standard normal cumulative distribution function, g and d are shape parameters, x is the location parameter, l is the scale parameter, and g is one of the following transformations:

$$g(x) = \begin{cases} \log(x) & \text{for the lognormal family} \\ \sinh^{-1}(x) & \text{for the unbounded family} \\ \log[x/(1-x)] & \text{for the bounded family} \\ x & \text{for the normal family.} \end{cases}$$

Nelson and Yamnitsky (1998) state that the appropriate transformation is chosen by estimating the skewness and kurtosis from a random sample $X_1, X_2, \ldots, X_n$ and by finding the unique Johnson cdf that matches the pair. One method for fitting target distributions from Johnson's translation system is via least-squares estimation, which is implemented in a software program called FITTR1 developed by Swain, Venkatraman and Wilson (1988). We applied the FITTR1 program to fit to Johnson distribution. Yet a Johnson distribution is not flexible enough to fit our noise distribution.

Univariate Bezier distributions provide an even more flexible alternative to standard distributions (Wagner and Wilson, 1996). Wagner and Wilson (1996) state that the univariate Bezier distribution is a special case of a spline curve and is constructed by fitting a curve to a specified number of points called control points. The control points are not data points; instead, they act as anchors for the Bezier cumulative distribution function and can be moved so as to alter the shape of the distribution.

A Bezier distribution with $n + 1$ control points is defined as

$$\mathbf{P}(t) = \begin{pmatrix} x(t) \\ F_X(x(t)) \end{pmatrix} = \sum_{i=0}^{n} B_{n,i}(t)\mathbf{p}_i$$

for $t \in [0,1]$, where $B_{n,i}(t)$ is the Bernstein polynomial (Bernstein, 1912)

$$B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-1}$$

for $t \in [0,1]$, $\mathbf{p}_i = (x_i, z_i)$ is i^{th} control point for i =0, 1, …, n.

At any value of t in the interval [0,1], the value of the Bezier distribution is simply a weighted average of the control points. To observe this, notice that the Bezier distribution can be written as

$$\mathbf{P}(t) = \begin{pmatrix} x(t) \\ F_X(x(t)) \end{pmatrix} = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-1} \begin{pmatrix} x_i \\ z_i \end{pmatrix} \mathbf{p}_i , \tag{5.1}$$

for $t \in [0,1]$. For any fixed value of t the sum of the weights is 1.

We use **PRIME** software to fit the noise distribution (Wagner and Wilson, 1996). **PRIME** is a software tool used to construct univariate Bezier distributions with or without data. It is an interactive, graphical software program that runs on a PC under Windows.

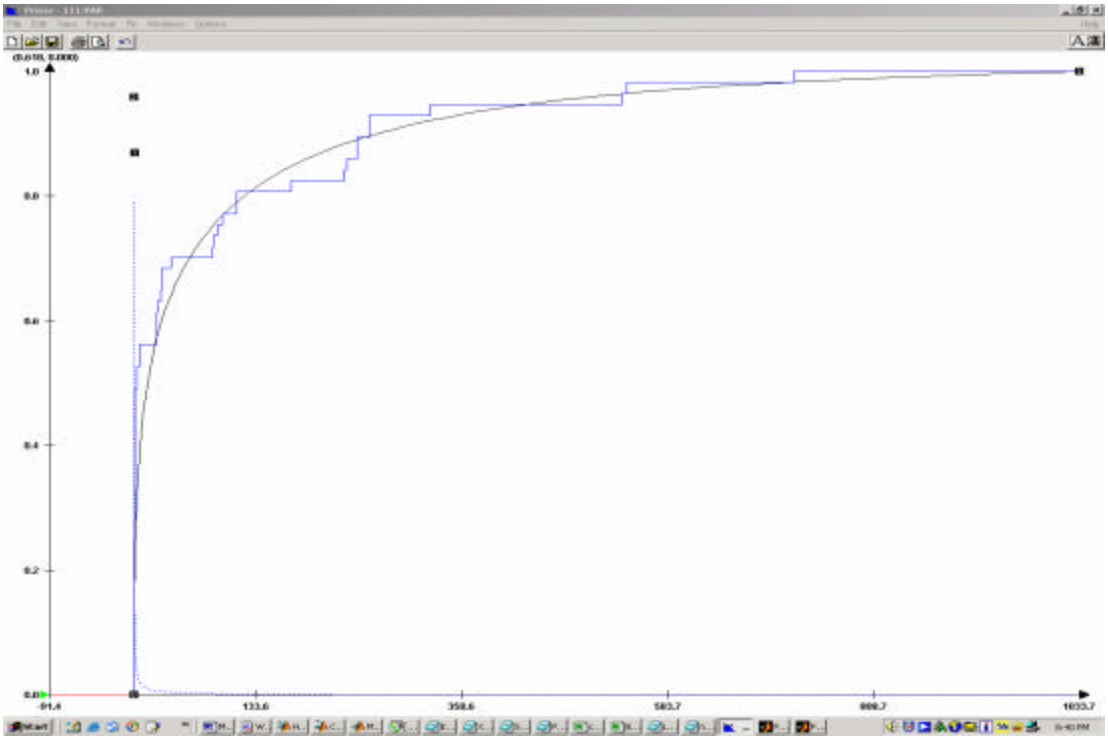


Figure 5.4 The Empirical CDF of the Noise_data vs. Fitted Bezier Distribution

A Bezier distribution with four control points shows a pretty good fit, as shown in Figure 5.4. The step function is the empirical cumulative distribution function from the Noise_data in Figure 5.2, and the smooth line is the fitted cumulative distribution function with control points (0,0), (0,0.87), (0.0.96), and (1034,1).

For the purpose of verifying the noise distribution pattern, we picked another data set from Wednesday of Week 5 of MIT Lincoln Lab PASCAL BSM files. The new sample data consist of traffic data information 10,000 seconds and have 66 non-zero noise data points. The average was 61.8 and the standard deviation was 122.8. Mean value and standard deviation of the new noise distribution are smaller than the previous Noise distribution. However, the standard deviation is still around twice as large as the mean, and the distribution shape was almost identical. Truly, the nonparametric Log-Rank test and Wilcoxon test for comparisons of the two noise distribution functions were performed and the tests of the similarity were not rejected at significance level of 5% (p-value = 0.40). The results are in Appendix 5.

5.2 RENDERING METHOD OF SIMULATION DATA

The simulated traffic data with an attack are the sum of three components: cycle, noise, and signal data. In other words, at time index t,

$$Traffic(t) = Cycle(t)+Noise(t)+Signal(t).$$

The cycle and noise represent the distribution of the system without attack, and the signal represents the attack component.

In this simulation study, the cycle, noise and signal are generated mutually independently with the same time span of 100 minutes or 6,000 seconds. Thus the simulated traffic data are also created for 6,000 seconds or 100 minutes.

5.2.1 CYCLE Background

Cycle data are the main background traffic data of the simulation study. In Section 5.1.1, we found that cycle data have 60-second cycle times with a total of 300 events, and they can be split into two with an interval of one second. The split probability is estimated from the Pure_sample_data as roughly 0.2.

There are two components in simulation of cycle data.

1. A random distribution of cycle peak or height with 60 seconds cycle time.
2. A random Bernoulli distribution to describe the possibility of split with parameter (p_s)

For simplicity, we assume that the cycle data have a constant height peak of 300. Actually, in the real cycle data, the heights were always very close to 300 with very little deviation. In the simulation studies, we tested three levels of split probability: 10% (Cycle 1), 20% (Cycle 2), and 40% (Cycle 3).

Cycle 2 is similar to the Pure_sample_data because the Pure_sample_data also have an approximately 20% split probability. Splits are least frequent in Cycle 1 and most frequent in Cycle 3.

If a split occurs, then the next cycle will be placed exactly 60 seconds later from the time when the previous cycle is completed. For example, if a cycle occurs without a split at 121 seconds, the next cycle will be placed at 181 (121+60) seconds. On the other hand,

if a cycle occurs with a split at 121 seconds, then the cycle is completed at 122 seconds, which is one second later from 121 seconds and the next cycle is placed at 182 (122+60) seconds. This actually happened in real data.

The split amount is determined as follows:

$$Split = \begin{cases} \frac{300U}{p_s}, & U < p_s \\ 0, & U \geq p_s \end{cases}$$

where, U is a random variable from uniform (0,1) distribution. Thus, the split amount is uniformly distributed.

We set the first cycle to occur at 31 seconds in our simulation. 31 seconds is an arbitrary value, and it is the mid-point of the first batch.

5.2.2 Noise

Noise is the main component of traffic data. In Section 5.1.2, we found that a Bezier distribution is a good fit to noise distribution. To generate noise data, we divide into two components:

1. Frequency: how frequently noise distribution occurs.
2. Distribution of noise data.

In the simulation studies, we used two levels of noise frequency. Noise 1 has a 0.5% probability of occurrence (p_o) and Noise 2 has a 5% probability of occurrence (p_o), which is 10 times more frequent than Noise 1. Noise 1 is similar to the Pure_sample in Figure 5.2.

For the distribution of noise data, we simply used the model found in Section 5.1.2: Bezier distribution with control points (0,0), (0,0.87), (0.0.96), and (1034, 1).

We generated noise data with the following steps:

1. Generate $U_1 \sim Uniform(0,1)$
2. If $U_1 < p_o$, generate $U_2 \sim Uniform(0,1)$
3. Find t^*, the solution to the equation $F_X(x(t)) = U_2$ where $F_X(x(t))$ is defined in Equation (5.1)
4. Compute $x(t^*)$ from Equation (5.1)

Step 3 can be done as a simple search method such as the bisection method due to the nondecreasing and monotonic properties of the cumulative distribution function (Cdf).

5.2.3 SIGNAL

We tested 3 types of signals:

1. Big bump (Signal 1)
2. Small bump (Signal 2)
3. Trend (Signal 3)

We assumed that signal data are normally distributed (Ye, 2003). For Signal 1, the mean is 900 and the standard deviation is 90 (an arbitrarily chosen 10% of mean value). For Signal 2, the mean is 300 and the standard deviation is 30. Signal 3 starts from the value of 1 and increases by one every second and ends at a value of 600. Figure 5.5 shows an example of each signal.

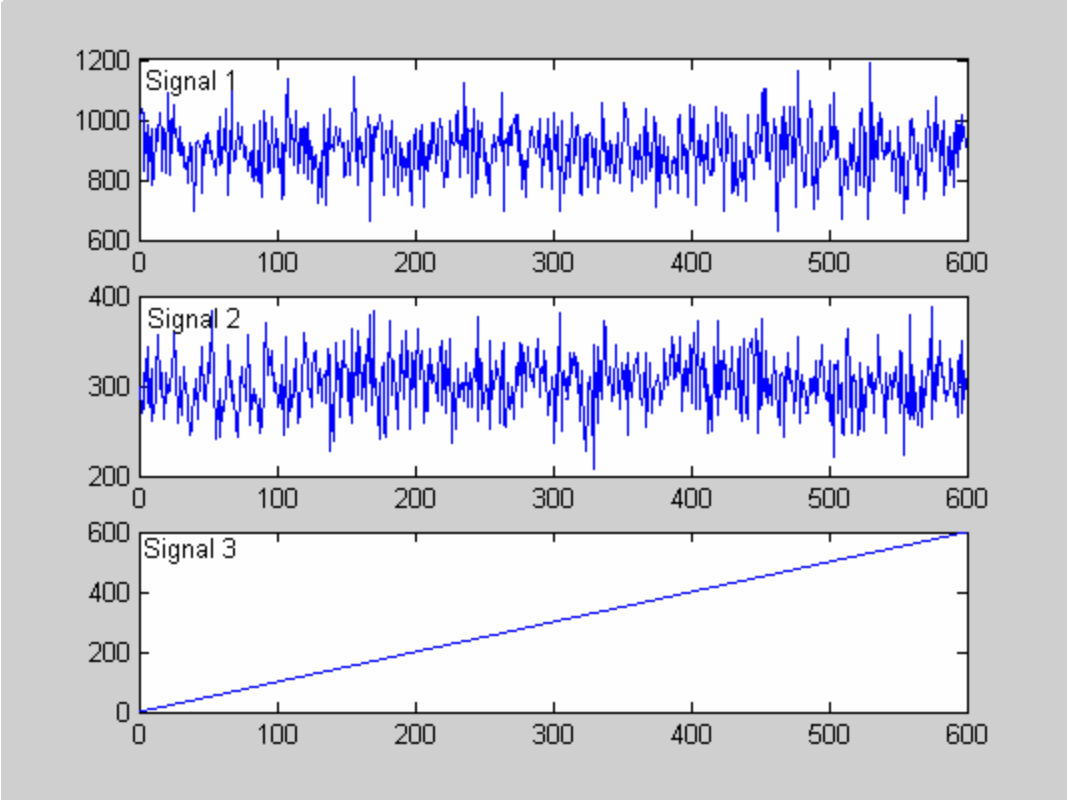


Figure 5.5 Example plots of Signal 1, Signal 2, and Signal 3

5.3. SIMULATION SCENARIOS

With three cycles, two noises, and three signal types, we have six scenarios without a signal that represent in-control data and 18 scenarios with a signal that generates out-of-control data. Table 5.1 summarizes the details of each data type, and Table 5.2 shows all possible scenarios with signal. These 18 scenarios along with six scenarios will be tested with three new and traditional SPC methods in Chapter 6.

Table 5.1 Simulation Data Definition

Cycle	Cycle 1	Peak=300, Split Probability = 0.1.
	Cycle 2	Peak=300, Split Probability = 0.2.
	Cycle 3	Peak=300, Split Probability = 0.4.
Noise	Noise 1	Mean Magnitude=80, Occurrence Probability = 0. 5%.
	Noise 2	Mean Magnitude=80, Occurrence Probability = 5%.
Signal	Signal 1	N $(900,90^2)$.
	Signal 2	N $(300,30^2)$.
	Signal 3	Linear with a slope of 1 and an intercept of 1.

Table 5.2 Simulation Combinations

Noise \ Cycle	Cycle1	Cycle2	Cycle3
Noise1	❑ Signal 1 ❑ Signal 2 ❑ Signal 3	❑ Signal 1 ❑ Signal 2 ❑ Signal 3	❑ Signal 1 ❑ Signal 2 ❑ Signal 3
Noise2	❑ Signal 1 ❑ Signal 2 ❑ Signal 3	❑ Signal 1 ❑ Signal 2 ❑ Signal 3	❑ Signal 1 ❑ Signal 2 ❑ Signal 3

APPENDIX 5. TEST OF SIMILARITY BETWEEN NOISE DISTRIBUTIONS

Distribution Analysis: Noise1

Variable: Noise1

Censoring Information	Count
Uncensored value	66

Nonparametric Estimates

Characteristics of Variable

Mean(MTTF)	Standard Error	95.0% Normal CI Lower	Upper
61.8333	15.1130	32.2123	91.4543

Median = 2
IQR = 76 Q1 = 1 Q3 = 77

Kaplan-Meier Estimates

Time	Number at Risk	Number Failed	Survival Probability	Standard Error	95.0% Normal CI Lower	Upper
1	66	23	0.651515	0.0586519	0.536559	0.766471
2	43	12	0.469697	0.0614326	0.349291	0.590103
4	31	3	0.424242	0.0608352	0.305008	0.543477
6	28	3	0.378788	0.0597099	0.261759	0.495817
21	25	1	0.363636	0.0592126	0.247582	0.479691
24	24	1	0.348485	0.0586519	0.233529	0.463441
25	23	2	0.318182	0.0573324	0.205812	0.430551
26	21	1	0.303030	0.0565689	0.192157	0.413903
29	20	2	0.272727	0.0548202	0.165282	0.380173
31	18	1	0.257576	0.0538278	0.152075	0.363076
77	17	1	0.242424	0.0527508	0.139035	0.345814
92	16	2	0.212121	0.0503211	0.113494	0.310749
94	14	1	0.196970	0.0489546	0.101020	0.292919
98	13	1	0.181818	0.0474757	0.088767	0.274869
100	12	1	0.166667	0.0458735	0.076756	0.256577
113	11	1	0.151515	0.0441345	0.065013	0.238017
171	10	1	0.136364	0.0422418	0.053571	0.219156
193	9	1	0.121212	0.0401738	0.042473	0.199951
244	8	1	0.106061	0.0379017	0.031775	0.180347
258	7	2	0.075758	0.0325712	0.011919	0.139596
274	5	1	0.060606	0.0293704	0.003041	0.118171
300	4	1	0.045455	0.0256398	0.000000	0.095708
349	3	1	0.030303	0.0211003	0.000000	0.071659
414	2	1	0.015152	0.0150363	0.000000	0.044622
667	1	1	0.000000	0.0000000	0.000000	0.000000

Distribution Analysis: Noise2

Variable: Noise2

```
Censoring Information  Count
Uncensored value          57


Nonparametric Estimates


Characteristics of Variable

            Standard   95.0% Normal CI
Mean(MTTF)     Error    Lower    Upper
   81.4035   20.0059  42.1928  120.614

Median = 4
IQR = 91  Q1 = 1  Q3 = 92


Kaplan-Meier Estimates

      Number
          at  Number     Survival   Standard    95.0% Normal CI
Time    Risk  Failed  Probability      Error     Lower     Upper
   1      57      16     0.719298  0.0595168  0.602647  0.835949
   2      41      12     0.508772  0.0662164  0.378990  0.638554
   4      29       2     0.473684  0.0661348  0.344062  0.603306
   6      27       2     0.438596  0.0657253  0.309777  0.567416
  25      25       3     0.385965  0.0644812  0.259584  0.512346
  26      22       1     0.368421  0.0638923  0.243194  0.493648
  30      21       1     0.350877  0.0632126  0.226983  0.474772
  31      20       2     0.315789  0.0615682  0.195118  0.436461
  41      18       1     0.298246  0.0605958  0.179480  0.417011
  86      17       1     0.280702  0.0595168  0.164051  0.397353
  87      16       1     0.263158  0.0583254  0.148842  0.377474
  92      15       1     0.245614  0.0570146  0.133868  0.357361
  98      14       1     0.228070  0.0555758  0.119144  0.336997
 113      13       2     0.192982  0.0522713  0.090533  0.295432
 172      11       1     0.175439  0.0503775  0.076701  0.274177
 230      10       1     0.157895  0.0482980  0.063232  0.252557
 233       9       1     0.140351  0.0460077  0.050177  0.230524
 244       8       2     0.105263  0.0406489  0.025593  0.184933
 258       6       2     0.070175  0.0338342  0.003862  0.136489
 324       4       1     0.052632  0.0295764  0.000000  0.110600
 534       3       1     0.035088  0.0243716  0.000000  0.082855
 539       2       1     0.017544  0.0173893  0.000000  0.051626
 721       1       1     0.000000  0.0000000  0.000000  0.000000
```

Distribution Analysis: Noise2, Noise1

```
Comparison of Survival Curves


Test Statistics

Method    Chi-Square  DF  P-Value
Log-Rank    0.699655   1    0.403
Wilcoxon    0.674280   1    0.412
```

CHAPTER 6

NEW SPC METHOD AND SIMULATION STUDIES

In this chapter, we describe the simulation results from various SPC intrusion detection approaches and compare the performance of each method. In addition to the methods described in Chapter 4, we also consider a new SPC method which will be described in Section 6.1. The simulation studies were done based on the simulation model described in Chapter 5.

In Section 6.1, we describe a modified batch mean method that was developed based on the regular batch mean method. We also introduced the concept of the modified batch mean and its properties. In Section 6.2 through 6.4 we provide simulation results for three different studies: a simulation study using standard control limits, a simulation study for performance comparisons of different charts, and a simulation study of robust control charts.

6.1 A MODIFIED BATCH MEAN CHART

6.1.1 The Definition of Modified Batch Mean

In the regular batch mean chart (RBM chart) we calculated the batch mean only at the end of each batch, then plot the batch mean statistic. We triggered an out-of-control signal when the statistic goes outside the control limits. A signal is triggered only at the end of each batch. When a large batch size is involved, the procedure becomes inefficient

when the signal occurs at the beginning or early stage of the batch. To rectify this problem we propose a modified batch mean (MBM) method. In the modified batch mean method we calculate a monitoring statistic at every elemental time unit, then plot it under the same control limits. In other words, we will plot one charting statistic for each batch in the regular batch mean methods, but will plot *b* charting statistics for each batch in the modified batch mean method, where *b* is the batch size.

We denote the t^{th} regular batch mean with batch size b as $\overline{Y}_t$ and the modified batch mean at the i^{th} time unit of the t^{th} batch as $\overline{Y}_{t(i)}$, i.e.,

$$\overline{Y}_t = \frac{\sum_{j=1}^{b} X(j)}{b}$$

and

$$\overline{Y}_{t(i)} = \frac{\sum_{j=1}^{i} X(j)}{b},$$

when *X (j)* is the j^{th} observation in the batch.

Figure 6.1 illustrates how $\overline{Y}_{t(j)}$ relates to $\overline{Y}_t$ and how it related to the raw data *X (j)*. In Figure 6.1, the elemental raw data (*X (j)*) were generated from IID normal distribution with mean 4 and variance 1 and the batch size is 60. The raw data plot shows the center line is around 4 and the modified batch mean graph shows the statistics increase as the time unit increases from 1 to 60. The increasing pattern happens as the raw data are positive. More discussion on this will be given in Chapter 8. At time unit 60, the MBM (60) is equal to the regular batch mean value (RBM) of the 60 elemental raw data.

1st batch	2nd batch		
x(1),x(2),…,x(b)	x(b+1),x(b+2),		

$$RBM = \frac{\sum_{j=1}^{b} x(j)}{b} \qquad MBM(i) = \frac{\sum_{j=1}^{i} x(j)}{b},\ i = 1,\ldots,b$$

$$\begin{aligned} RBM &= MBM(1) + \frac{x(2)+\cdots+x(b)}{b} \\ &= MBM(2) + \frac{x(3)+\cdots+x(b)}{b} \\ &\vdots \\ &= MBM(b) \end{aligned}$$

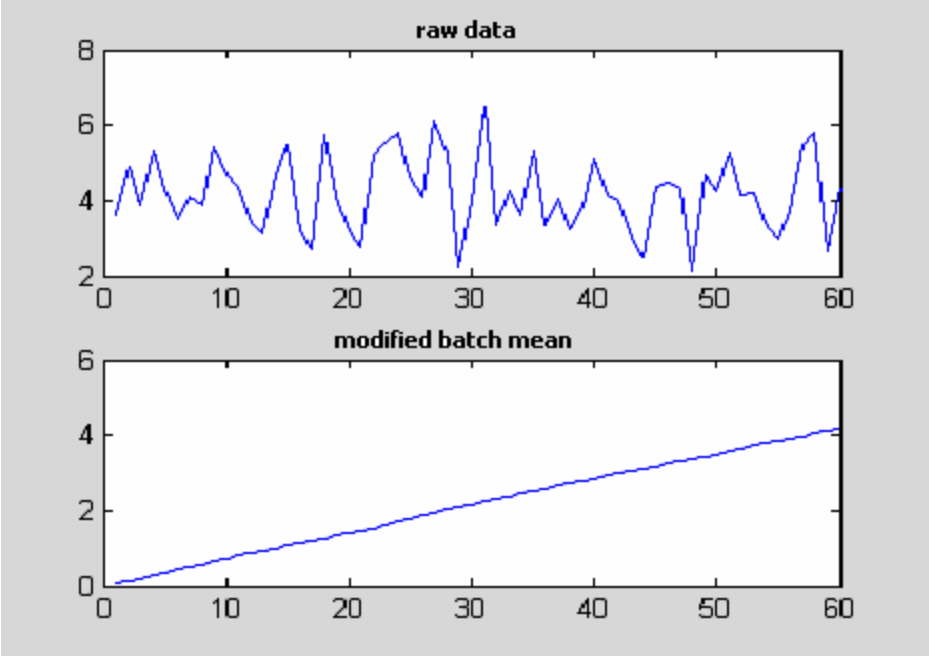


Figure 6.1 The Concept of Modified Batch Mean

6.1.2 The Properties of Modified Batch Mean

For the modified batch mean methods to work, we need the following assumptions:

1. Original (or elemental) data can be gathered by in a sequential manner.

2. Detection of one-sided shifts is the objective.

Note that the modified batch mean equals the regular batch mean at the end of the batch. If there is a signal that will be detected by the regular batch mean chart at the end of the batch, i.e. *RBM > control limit*, as a consequence, the same signal will be detected by the MBM chart earlier than or at the same time as the RBM Chart.

If the size of the signal is large and it happens at an early stage of the batch, the signal may be detected early as the *MBM* statistic quickly becomes bigger than the control limit. If the size of the signal is moderate, the signal may not be detected until the end of the batch.

In any case when there is a signal, the MBM chart will detect the signal either earlier than or at the same time as the regular batch means chart. Note that the possibility that the MBM chart will detect the signal earlier can increase as the size of the signal increases. Also, the impact to the ARL_1 or detection time can be quite significant as ARL_1 is relatively small for large signals.

Figure 6.2 shows the detection time of a MBM chart. For illustration purposes, we generated an in-control data (no attack) from IID normal distribution with mean 10 and variation 1. For these data, we added signals of mean shifts with various shift magnitudes (m= 1, 10, 50, and 200).

We chose the batch size as 60 and the control limit is 3-s level of batches.

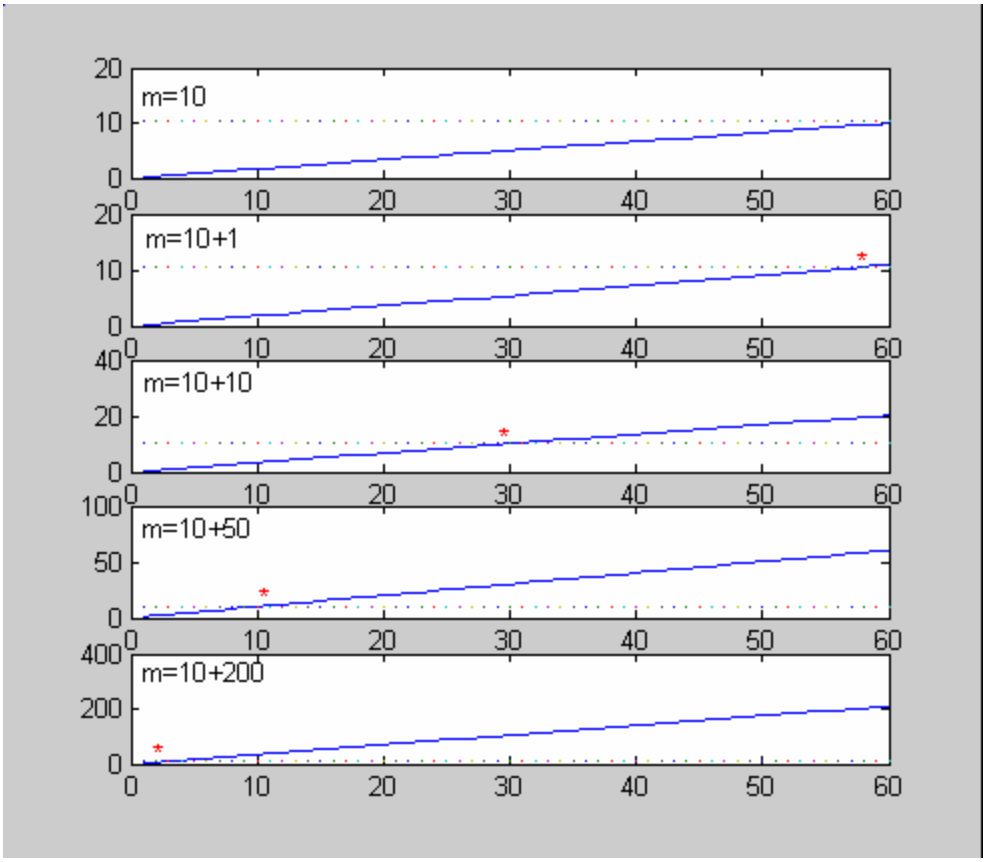


Figure 6.2 Illustration of MBM ARL_1

The detection time is faster if the signal is larger. The red asterisk marks the detection point in Figure 6.2. If we used a regular batch mean chart, then we would detect the signal at the end of the batch, i.e., at the 60^{th} second.

On the other hand, if there is no signal, it is still possible that the regular batch mean chart will trigger the alarm (i.e., a false alarm) when *RBM > control limit*. Similarly, the MBM chart will trigger the false alarm earlier than or at the same time as the RBM chart. However, the chance that the MBM chart will trigger the alarm earlier is small because there is no signal. Also, even if the MBM chart does trigger the alarm earlier, the impact to the ARL_0 is small because ARL_0 is relatively large.

6.1.3 The Variants of Modified Batch Mean Chart

The modified batch mean concept can be applied to any control chart that involves batch mean. Below we explain the detailed implementation of applying the MBM concept to the Shewhart chart, the Cusum chart, and the EWMA chart described in Chapter 3.

Modified Batch Mean Shewhart Chart

The modified batch mean (MBM) Shewhart chart is defined as the regular batch mean chart with the regular batch mean replaced by the modified batch mean.

Define $\overline{Y}_t$ to be the regular batch mean of the t^{th} batch and $\overline{Y}_{t(j)}$ to be the modified batch mean of the j^{th} time unit of the t^{th} batch. The formulas are in Table 6.1.

Table 6.1 MBM Shewhart Chart

Chart	Monitoring Statistic	Control Limit (CL)
Regular	$\overline{Y}_t = \sum_{i=b\cdot(t-1)+1}^{bt} X(i)/b$, for batch t	If $\overline{Y}_t > CL$, raise alarm at the end of batch t.
Modified	$\overline{Y}_{t(j)} = \sum_{i=b\cdot(t-1)+1}^{b(t-1)+j} X(i)/b$, for batch t, j=1, …, b	If $\overline{Y}_{t(j)} > CL$, raise alarm at the j^{th} time unit of batch t.

Modified Batch Mean Cusum Chart

The modified batch mean Cusum chart is the Cusum chart applying the modified batch mean. The formulas are given in Table 6.2.

Table 6.2 Monitoring Statistics of MBM Cusum Chart

Chart	Monitoring Statistic	Control Limit
Regular	$C_t = \max[0, \overline{Y}_t - (m_0 + K) + C_{t-1}]$	If $C_t > H$, raise alarm at the end of batch t.
Modified	$C_{t(j)} = \max[0, \overline{Y}_{t(j)} - (m_0 + K) + C_{t-1}]$	If $C_{t(j)} > H$, raise alarm at the j^{th} time unit of batch t.

Modified Batch Mean EWMA Chart.

The modified batch mean EWMA chart is the EWMA chart applying the modified batch mean. The formulas are given in Table 6.3.

Table 6.3 Monitoring Statistics of MBM EWMA Chart

Chart	Monitoring Statistic	Control Limit(CL)
Regular	$z_t = l\,\overline{Y}_t + (1-l)z_{t-1}$	If $z_t > CL$, raise alarm at the end of batch t.
Modified	$z_{t(j)} = l\,\overline{Y}_{t(j)} + (1-l)z_{t-1}$	If $z_{t(j)} > CL$, raise alarm at the j^{th} time unit of batch t.

Although the MBM concept is expected to improve the regular batch mean chart, it is not clear how big the impact is and under what situations this will happen. To understand and compare the performance of the MBM chart as well as the RBM charts described in Chapter 4, we conducted three parts of simulation studies based on the simulated model developed in Chapter 5.

6.2 SIMULATION STUDY USING STANDARD CONTROL LIMITS

The objective of this simulation study is to investigate the performance of the MBM charts under the control limits recommended by the standard textbooks in SPC.

Standard control limits are defined to be the control limits based on the parameters recommended by IID normal data. These limits can be found in standard statistical textbooks, such as Montgomery (2001). The typical parameters for the three common SPC charts are as follows.

- Parameters of standard control limit for $ARL_0 = 370$.
  - Shewhart; c = 3.0, where c is the control limit parameter.
  - Cusum; k = 0.5, h = 4.77, where k is the chart parameter and h is the control limit parameter.
  - EWMA; λ=0.2, L=2.806, where λ is the chart parameter and L is the control limit parameter.

6.2.1 Simulation Settings

The simulation settings for using standard control limits are as follows. The simulation factors are cycle, noise, batch size, and signal. As described in Chapter 5, we considered three levels (Cycle 1, Cycle 2, and Cycle 3) are used for the cycle factor, two levels (Noise 1 and Noise 2) are used for the noise factor, and three levels (Signal 1, Signal 2, and Signal 3) are used for the signal factor.

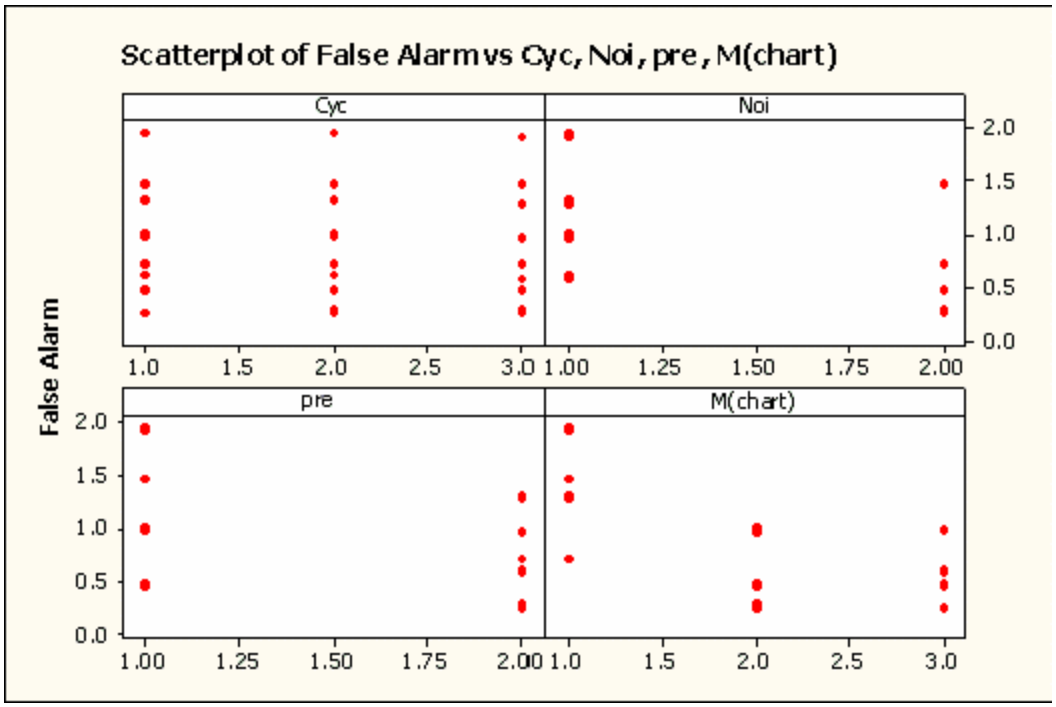
6.2.2 Simulation Results

The simulation results using standard control limits are summarized in Appendix 6.A. The performance measures were the average number of false alarms and average detection times in 10,000 runs. As defined in Chapter 3, a false alarm is an alarm when there is no signal (attack). So the number of false alarms was recorded during the no signal period (from 1 to 5,400 seconds) in each run. Then we computed the average value of the number of false alarm based on 10,000 simulations. In the same way, the detection times were recorded during the attack period (from 5,401 to 6,000 seconds) in 10,000 runs and their average values were computed.

Figure 6.3 shows the scatter plots of performance measures versus factors. “M” refers to the charting methods (M = 1: MBM Shewhart chart, M = 2: MBM Cusum chart, and M = 3: MBM EWMA chart). “pre” refers to the batch sizes (pre = 1: 60 seconds batch size and pre = 2: 120 seconds batch size).

According to Figure 6.3, we observed the following:

1. Noise 2 affected detection time and the occurrence of false alarms. Noise 1 had quicker detection time but more false alarms than Noise 2.
2. Charting methods had an effect on false alarm occurrence and on detection time. For example, the MBM Shewhart chart had the most false alarms among the three charts. And on average, the MBM Cusum chart had the fewest false alarms but also had the longest detection time.



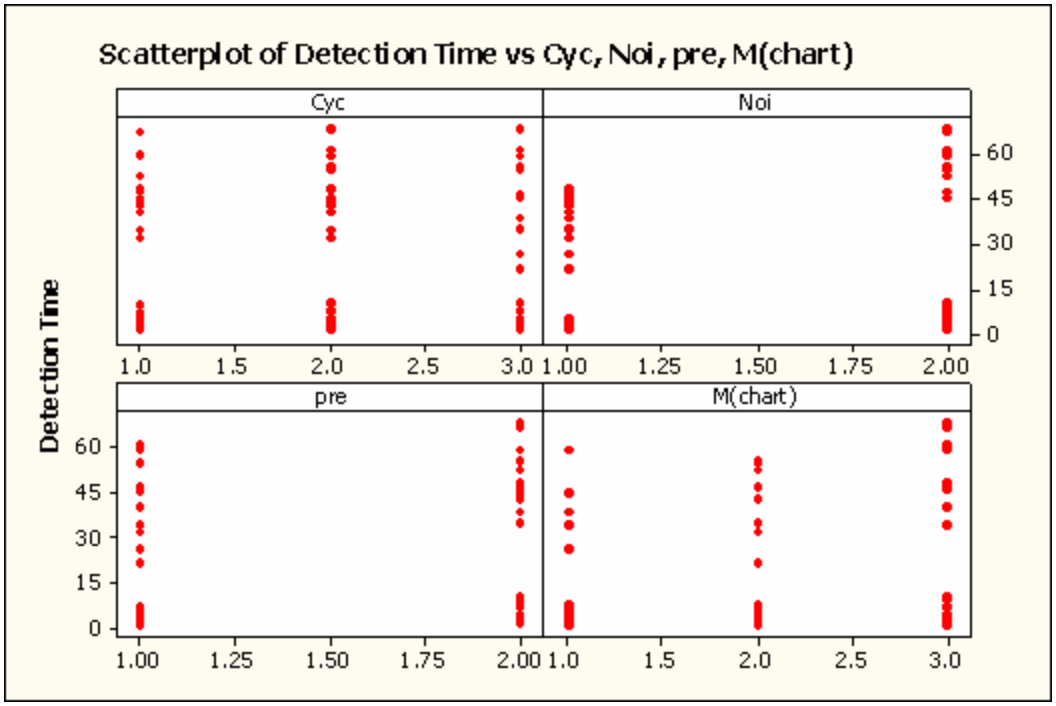


Figure 6.3 The Relationships between the False Alarm and Input Factors.

3. Signal 1 had the shortest detection time i.e., the signal was detected earliest, which was expected because Signal 1 had the largest magnitude of signal height.
4. Preprocessing affected the number of false alarms and detection time. The batch size of 120 seconds had fewer false alarms but a longer detection time.
5. Cycle had no effect on either the number of false alarms or on detection time.
6. Number of false alarms and detection time were negatively correlated, i.e., if the control limit was narrow, then we had high false alarm rate and fast detection; if the control limit was wide, then the false alarm rate was low and detection was slow.

To understand how the simulation factors affect the performance of the charts, analysis of variance (ANOVA) with 4 factors (cycle (cyc), noise (noi), signal (sig), and preprocessing batch size (pre)) were applied to each of the performance measures. The results of ANOVA were summarized in Appendix 6.B.

It was found that the cycle factor had no effect on either the average number of false alarms or average detection time, but there were significant interaction effects among various factors.

With two performance measures involved, it is difficult to conclude that one chart is better than the other when the two performance measures are in conflict. A fair comparison would be to first fix one performance measure (the false alarm) then compare the other performance measure (the detection time). This will be done in the next section.

6.2.3 Lessons Learned

The simulations above were done using standard control limits with the parameters obtained from standard textbooks. Note that under standard control limits and IID normal data, the expected ARL_0 should be 370 runs when we use the parameters as two-sided control limits or should be 740 runs when we use the parameters as a one-sided control limit. In our studies we focused on detecting one-sided shifts, thus the ARL_0 should be 740 runs.

For a total number of 5,400 simulated seconds, the expected numbers of false alarm equals 5,400/(60*740) = 0.1216 for a batch size of 60 seconds, and 5,400/(120*740) = 0.0608 for a batch size of 120 seconds.

However, as shown in Appendix 6.A, the observed average numbers of false alarms are much higher than expected. For example, the MBM Shewhart chart had an average number of false alarm = 1.96 for Cycle 1 and Noise 1 case. This value is 16 times larger than the expected value of 0.1216. In other words, this implies that the corresponding ARL_0 equals 740/16 = 46.25, much less than 740. This could be caused by violation of the assumptions that the data are identically and independently normally distributed.

To find out what assumptions may have been violated, Normal probability plot and correlation tests were applied. It was found that the main cause was because of the violation of the normality assumption. As shown in the simulation results (Appendix 6.A), all the MBM charts were sensitive to violation of the normality assumption and the MBM Shewhart chart was the mostly sensitive of the three charts.

6.3 SIMULATION STUDY ON PERFORMANCE COMPARISON OF DIFFERENT CHARTS

As pointed out in the last section, it is difficult to compare the performance of different charts when two performance measures are not pointing in the same direction. A common way to compare the performance of SPC charts is to fix the average number of false alarms, or the in-control average run length (ARL_0), then compare the average detection time or the out-of-control average run length (ARL_1). As shown in Section 6.2, the standard control limits will result in different ARL_0, thus one needs to adjust the control limits so that the ARL_0 is fixed. We denote these adjusted control limits to be the actual control limits.

Below, we investigate the performance of different charts with an actual $ARL_0 = 370$. For a fair comparison of performance, we set the actual control limits that give $ARL_0 = 370$ minutes for each chart and each scenario combination.

6.3.1 Simulation Settings and Actual Control Limits

In this simulation study we restricted the simulation parameters as noise and signal. The noise factor had two levels (Noise 1 and Noise 2), and the signal factor had three levels (Signal 1, Signal 2, and Signal 3). The cycle factor was removed because, as shown in Section 6.2, it had no effect on detection time or on false alarms. The batch size factor was also removed to restrict the batch size to 60 seconds.

To determine the actual control limits, simulations with 36 million runs were used in each scenario. In the simulation study we considered the one-sided control limit because

we are interested only in detecting the increase of event intensity due to an intrusion attack. Table 6.4 shows the actual control limit for each scenario.

Table 6.5 shows the simulated in-control ARL (ARL_0) obtained based on simulations with the control limits given in Table 6.4. The standard error (s.e.) values were calculated and displayed in parenthesis, i.e., $s.e. = \sqrt{\frac{s.d.}{n}}$, where *s.d.* is the sample standard deviation of run length and *n* is the observed number of run lengths.

Table 6.4 Actual Control Limits for Simulation Data (One-sided ARL_0=370, Batch Size=60)

Control Limits		Shewhart c	Cusum(k=0.5) h	EWMA(l =0.2) L
Standard Limits		2.782	4.10	2.601
Actual Limits	Noise 1	8.349	8.820	5.08
	Noise 2	4.615	6.027	3.404

Table 6.5 One-sided ARL_0 with Actual Control Limits (Batch Size=60)

ARL_0 (standard error)		Shewhart	Cusum (k=0.5)	EWMA (l =0.2)
Actual Limits	Noise1	371.35 (9.32)	370.29 (9.12)	370.06 (9.68)
	Noise2	370.27 (8.74)	370.01 (9.14)	370.31 (9.47)

6.3.2 Simulation Results

The simulation results using the actual one-sided control limits of the 18 scenarios are given in Table 6.6. The average detection times (in seconds) are summarized in Table 6.7 for easier comparison.

In Table 6.6, the in-control average run times (ART_0) are displayed and were calculated by multiplying the batch size to ARL_0. ART_0 were used instead of ARL_0 because the batch size may vary for different charts. For the MBM charts, the batch size equals 60 seconds. The time unit of ART_0 is seconds.

Table 6.6 Detection Time Using Actual Control Limits

noi	sig	M	NAME	ART_0	Modified DT	Regular DT
1	1	1	a_mbm60_111_mean	22160	**2.00**	60
1	2	1	a_mbm60_112_mean	22160	**4.19**	60
1	3	1	a_mbm60_113_mean	22160	**46.74**	60
2	1	1	a_mbm60_121_mean	22182	**2.99**	60
2	2	1	a_mbm60_122_mean	22182	**6.94**	60
2	3	1	a_mbm60_123_mean	22182	**54.19**	60
1	1	2	a_cusum60_111_mean	22323	**1.98**	60
1	2	2	a_cusum60_112_mean	22323	**3.81**	60
1	3	2	a_cusum60_113_mean	22323	**35.14**	60
2	1	2	a_cusum60_121_mean	22201	**2.80**	60
2	2	2	a_cusum60_122_mean	22201	**5.84**	60
2	3	2	a_cusum60_123_mean	22201	**58.35**	60
1	1	3	a_ewma60_111_mean	22187	**1.99**	60
1	2	3	a_ewma60_112_mean	22187	**4.48**	60
1	3	3	a_ewma60_113_mean	22187	**47.05**	60
2	1	3	a_ewma60_121_mean	22189	**2.94**	60
2	2	3	a_ewma60_122_mean	22189	**7.92**	60
2	3	3	a_ewma60_123_mean	22189	**67.83**	106.45

Table 6.7 Detection Time Comparisons

		Detection Time					
		Shewhart_a		Cusum_a		EWMA_a	
Signal	**Noise**	Modified	Regular	Modified	Regular	Modified	Regular
sig1	noi1	2.00	60	1.98	60	1.99	60
sig2		4.19	60	3.81	60	4.48	60
sig3		46.74	60	35.14	60	47.05	63.7
sig1	noi2	2.99	60	2.80	60	2.94	60
sig2		6.94	60	5.84	60	7.92	60
sig3		54.19	60	58.35	60	67.83	106.45

According to Table 6.6, we observed the followings.

1. As expected, in all scenarios, each chart detected Signal 1 faster than either Signal 2 or Signal 3, because the magnitude of the signal affects detection time, and Signal 1 had the largest magnitude. Signal 1 was detected about twice as fast as Signal 2.
2. The noise factor also affected detection time. For Cusum and EWMA charts, if the noise was dense (Noise 2), then the detection time increased. In other words, the detection time for Noise 2 was longer than for Noise 1.
3. The MBM Cusum chart worked best for all cases except for the case with Noise 2 and Signal 3. In the case with Noise 2 and Signal 3, the MBM Shewhart chart had the quickest detection time of all the control charts.
4. It was apparent that the MBM chart was superior to the RBM charts for most cases. The RBM charts detected the signals at 60, but the MBM charts detected Signal 1 in about 2 seconds and detected Signal 2 in about 5 seconds, which is about 30 seconds, or 10 times, faster than the RBM charts. In the cases of Signal 3, the differences in detection time with the MBM chart and the RBM chart were smaller because the signal magnitude is smaller.
5. In summary, the MBM methods can significantly improve the performance of the RBM Charts when (i) the batch size is large, and (ii) the signal is large.

6.4 SIMULATION STUDY FOR ROBUST MBM CHARTS

As shown in Section 6.2, the performances (false alarm and detection time) of the RBM and MBM methods under standard control limits are quite different from the expected

performances. This difference is mainly caused by violation of the normality assumption of the charting statistic. To rectify this problem, some robust BM charts were developed as follows.

According to the Central Limit Theorem, one can increase the batch size so that the charting statistic can be better approximated to normal distribution. Alternatively, one can change the charting parameter (k value in Cusum charts or l value in EWMA charts) so that the actual chart performance is close to the expected performance.

Below we will consider three robust BM charts:

- Batch Mean with large batch size,
- Cusum with small k value,
- EWMA with small l value.

6.4.1 Robust Batch Mean Shewhart Chart

To compare how batch size affects the robustness property of the BM Shewhart chart, we first determined the one-sided control limits of the BM Chart with different batch sizes that will result in the same in-control average run time (ART_0) equal to 370 minutes. Table 6.8 displays these limits and the corresponding values of ARL_0 and ART_0.

Under the standard limits in Table 6.8, the in-control average run time (ART_0) is expected to be approximately equal to 370 minutes if the charting statistics can be well approximated by normal distribution. Table 6.9 shows the actual ART_0s based on simulations under the two noise conditions. Clearly, the ART_0 approached the expected ART_0 value of 370 minute as the batch size increased. The approximation was satisfactory when the batch size is increased to 40 minutes (cycles).

Table 6.8 Normal Recommended Control Limit for One-sided Shewhart Chart (ART_0 = 370 minute).

ARL_0 (in minute)	370	185	37	18.5	9.25
Batch size	1	2	10	20	40
ART_0 (in minute)	370	370	370	370	370
C	2.782	2.549	1.927	1.607	1.237

Table 6.9 ART_0 of Normal recommended Control limit

Batch size	C	ART_0 Noise1	ART_0 Noise2
B = 1 cycle	2.782	39.42	43.92
10 cycle	1.927	161.79	244.45
20 cycle	1.607	259.67	312.44
40 cycle	**1.237**	**344.55**	**354.14**

6.4.2 Robust Batch Mean Cusum Chart

Similar to the Robust BM Shewhart chart, one can develop a robust BM Cusum chart by choosing an appropriate value for the charting parameter k.

Table 6.10 shows different values of charting parameter k and the corresponding control limit h so that the expected ART_0 equal 370 minutes. The last two columns display the actual ART_0 obtained from simulations. It shows that the ART_0 approached the expected value of 370 minutes as k decreased and the approximation was satisfactory at k = 0.05.

Table 6.10 Normal Recommended Control Limit for One-sided Cusum Chart

k	h	ART_0 Noise1	ART_0 Noise2
0.5	4.10	70.21	104.33
0.25	6.72	127.09	197.23
0.1	10.72	278.57	301.73
0.05	**13.47**	**352.77**	**340.47**

6.4.3 Robust EWMA Chart

Montgomery (1999) proposed the idea of developing a robust EWMA chart by choosing the chart parameter l =0.01. Table 6.11 shows how the ART_0 changed as the value of the chart parameter l decreased.

Table 6.11 Normal Recommended Control Limit for One-sided EWMA Chart

Lambda(l)	L	ART_0 Noise1	ART_0 Noise2
0.2	2.601	75.36	118.49
0.1	2.402	115.11	180.93
0.05	2.142	185.03	251.14
0.01	**1.282**	**355.62**	**364.50**

6.4.4 Performance Comparison of Robust MBM Charts

As shown in the previous sections, different robust control charts could be developed by choosing the appropriate number of batch sizes in the Shewhart chart or by choosing the appropriate charting parameters in the Cusum chart or in the EWMA chart. These robust charts will produce an actual ART_0 approximately equal to the expected ART_0 when standard control limits in textbooks are used. Below we compare the detection time performance of these robust control charts as well as that of the control charts based on actual control limits described in Section 6.3.

To further understand how these charts perform under different signals, in addition to the three signal patterns described in Chapter 5, we further considered the following six signal patterns. We skipped Signal 3 as it is not a step jump signal. The signals are normally distributed and the standard deviation of each signal being 10% of the mean.

- Signal 4: Normal $(100,10^2)$
- Signal 5: Normal $(50, 5^2)$
- Signal 6: Normal $(20, 2^2)$
- Signal 7: Normal $(10, 1^2)$
- Signal 8: Normal $(5, 0.5^2)$
- Signal 9: Normal $(1, 0.1^2)$

The results of the actual MBM charts and the robust MBM charts for various signals were summarized in Table 6.12.

According to Table 6.12, we observed the following:

1. Overall, the MBM Shewhart charts performed worse than the MBM Cusum and EWMA charts.
2. The robust MBM Shewhart chart performed worse than the MBM Shewhart chart with an actual control limit for large signals, but the robust MBM Shewhart charter performed better for small signals.
3. The robust MBM Cusum chart had similar performance as the MBM Cusum chart with an actual limit for small signals, but the robust MBM Cusum chart performed worse for large signals (except for Signal 9).

Table 6.12 Summary of Detection time in Simulation

Chart		Control Charts with Actual Control Limits			Robust Control Charts with Standard Control Limits		
		Shewhart B=1	Cusum k=0.5	EWMA l =0.2	Shewhart B=40	Cusum k=0.05	EWMA l =0.01
ARL_0 (Min.)		369.65	372.41	370.11	344.55	355.14	355.62
ART_0 (Sec.)		22160	22323	22187	20591	21287	21315
Modified	DT_sig1	2.00	1.98	1.99	15.94	1.92	2.08
	DT_sig2	4.19	3.81	4.48	45.97	3.82	5.26
	DT_sig4	11.97	12.88	12.06	131.36	17.08	12.72
	DT_sig5	23.39	25.20	23.60	249.56	31.05	24.86
	DT_sig6	42.40	46.83	42.85	542.46	83.66	46.01
	DT_sig7	1,901.3	100.27	98.37	891.92	149.08	94.17
	DT_sig8	5,870.5	220.91	252.62	1,319.3	280.36	201.62
	DT_sig9	15,596.4	2,667.9	2,675.1	2,145.4	1,493.1	1,029.7
Regular	DT_sig1	60	60	60	2400	60	60
	DT_sig2	60	60	60	2400	60	60
	DT_sig4	60	60	60	2400	60	60
	DT_sig5	60	60	60	2400	60	60
	DT_sig6	60	60	60	2400	119.29	60
	DT_sig7	1,917.8	120	120	2400	177.00	120
	DT_sig8	5,888.6	235.30	277.31	2400	293.41	232.46
	DT_sig9	15,616.4	2,686.5	2,733.4	2400	1,511.1	1,066.1

4. The robust MBM EWMA chart had similar or better performance than that of the MBM EWMA chart with an actual limit.
5. Overall, the robust EWMA chart had the best performance in most cases. It was also easy to implement because the control limits could be determined as the standard limits from the textbook.

6.5 SUMMARY

In this chapter, we illustrated the modified batch mean (MBM) concept and the modified batch mean charts of Shewhart, Cusum and EWMA. The MBM charts outperform the RBM charts because they monitor at each time unit within the batch. The MBM charts can be significantly better under situations characterized by large batch size and large signals.

One difficulty of applying the regular or modified BM charts is that the control limits need to be determined by simulation. To rectify this problem, three robust MBM charts were developed. The robust MBM Shewhart chart was developed based on choosing appropriate values of batch size. The robust MBM Cusum chart and robust MBM EWMA chart were developed on choosing appropriate values of charting parameters. According to our simulation studies, the robust EWMA chart has the best detection time performance and its control limits can be easily determined. Thus it is the control chart of choice the SPC intrusion detection problem.

APPENDIX 6A. SIMULATION RESULTS USING STANDARD CONTROL LIMITS

Cyc	Noi	Sig	pre	m	Name	False Alarm	Modified DT	Regular DT
1	1	1	1	1	mbm60_111_mean	1.96	1	60
1	2	1	1	1	mbm60_121_mean	1.4682	1.99	60
1	1	2	1	1	mbm60_112_mean	1.96	2.61	60
1	2	2	1	1	mbm60_122_mean	1.4682	4.9704	60
1	1	3	1	1	mbm60_113_mean	1.96	34.6807	60
1	2	3	1	1	mbm60_123_mean	1.4682	45.2377	60
2	1	1	1	1	mbm60_211_mean	1.96	1	60
2	2	1	1	1	mbm60_221_mean	1.48	1.99	60
2	1	2	1	1	mbm60_212_mean	1.96	2.61	60
2	2	2	1	1	mbm60_222_mean	1.48	5.1865	60
2	1	3	1	1	mbm60_213_mean	1.96	34.6807	60
2	2	3	1	1	mbm60_223_mean	1.48	45.0646	60
3	1	1	1	1	mbm60_311_mean	1.9192	1	60
3	2	1	1	1	mbm60_321_mean	1.48	1.99	60
3	1	2	1	1	mbm60_312_mean	1.9192	2.54	60
3	2	2	1	1	mbm60_322_mean	1.48	5.1865	60
3	1	3	1	1	mbm60_313_mean	1.9192	26.4502	60
3	2	3	1	1	mbm60_323_mean	1.48	45.0646	60
1	1	1	2	1	mbm120_111_mean	1.31	2	120
1	2	1	2	1	mbm120_121_mean	0.7109	2.99	120
1	1	2	2	1	mbm120_112_mean	1.31	4	120
1	2	2	2	1	mbm120_122_mean	0.7109	7.5025	120
1	1	3	2	1	mbm120_113_mean	1.31	44.5177	120
1	2	3	2	1	mbm120_123_mean	0.7109	59.1446	120
2	1	1	2	1	mbm120_211_mean	1.31	2	120
2	2	1	2	1	mbm120_221_mean	0.72	2.99	120
2	1	2	2	1	mbm120_212_mean	1.31	4	120
2	2	2	2	1	mbm120_222_mean	0.72	8.0163	120
2	1	3	2	1	mbm120_213_mean	1.31	44.5177	120
2	2	3	2	1	mbm120_223_mean	0.72	59.0037	120
3	1	1	2	1	mbm120_311_mean	1.2844	1.92	120
3	2	1	2	1	mbm120_321_mean	0.72	2.99	120
3	1	2	2	1	mbm120_312_mean	1.2844	3.84	120
3	2	2	2	1	mbm120_322_mean	0.72	8.0163	120
3	1	3	2	1	mbm120_313_mean	1.2844	38.9062	120
3	2	3	2	1	mbm120_323_mean	0.72	59.0037	120
1	1	1	1	2	cusum60_111_mean	1.01	1.0168	60
1	2	1	1	2	cusum60_121_mean	0.4746	2.5952	60
1	1	2	1	2	cusum60_112_mean	1.01	2.9909	60
1	2	2	1	2	cusum60_122_mean	0.4746	5.0452	60
1	1	3	1	2	cusum60_113_mean	1.01	31.7108	60

1	2	3	1	2cusum60_123_mean	0.4746	46.8634	60
2	1	1	1	2cusum60_211_mean	1.01	1.0168	60
2	2	1	1	2cusum60_221_mean	0.46	2.6677	60
2	1	2	1	2cusum60_212_mean	1.01	2.9909	60
2	2	2	1	2cusum60_222_mean	0.46	5.2366	60
2	1	3	1	2cusum60_213_mean	1.01	31.7108	60
2	2	3	1	2cusum60_223_mean	0.46	54.6183	60
3	1	1	1	2cusum60_311_mean	0.9807	1.0336	60
3	2	1	1	2cusum60_321_mean	0.46	2.6677	60
3	1	2	1	2cusum60_312_mean	0.9807	2.9006	60
3	2	2	1	2cusum60_322_mean	0.46	5.2366	60
3	1	3	1	2cusum60_313_mean	0.9807	21.8488	60
3	2	3	1	2cusum60_323_mean	0.46	54.6183	60
1	1	1	2	2cusum120_111_mean	0.9799	1.9544	120
1	2	1	2	2cusum120_121_mean	0.2638	3.1035	120
1	1	2	2	2cusum120_112_mean	0.9799	4.1292	120
1	2	2	2	2cusum120_122_mean	0.2638	7.3876	120
1	1	3	2	2cusum120_113_mean	0.9799	42.556	120
1	2	3	2	2cusum120_123_mean	0.2638	52.3348	120
2	1	1	2	2cusum120_211_mean	0.9799	1.9544	120
2	2	1	2	2cusum120_221_mean	0.29	3.1348	120
2	1	2	2	2cusum120_212_mean	0.9799	4.1292	120
2	2	2	2	2cusum120_222_mean	0.29	7.8355	120
2	1	3	2	2cusum120_213_mean	0.9799	42.556	120
2	2	3	2	2cusum120_223_mean	0.29	55.2693	120
3	1	1	2	2cusum120_311_mean	0.9571	1.96	120
3	2	1	2	2cusum120_321_mean	0.29	3.1348	120
3	1	2	2	2cusum120_312_mean	0.9571	4.0171	120
3	2	2	2	2cusum120_322_mean	0.29	7.8355	120
3	1	3	2	2cusum120_313_mean	0.9571	35.2438	120
3	2	3	2	2cusum120_323_mean	0.29	55.2693	120
1	1	1	1	3ewma60_111_mean	0.9999	1.0842	60
1	2	1	1	3ewma60_121_mean	0.4628	2.6981	60
1	1	2	1	3ewma60_112_mean	0.9999	3.1425	60
1	2	2	1	3ewma60_122_mean	0.4628	6.6642	60
1	1	3	1	3ewma60_113_mean	0.9999	40.2991	60
1	2	3	1	3ewma60_123_mean	0.4628	60.0162	90.156
2	1	1	1	3ewma60_211_mean	0.9999	1.0842	60
2	2	1	1	3ewma60_221_mean	0.48	2.7071	60
2	1	2	1	3ewma60_212_mean	0.9999	3.1425	60
2	2	2	1	3ewma60_222_mean	0.48	7.0111	60
2	1	3	1	3ewma60_213_mean	0.9999	40.2991	60
2	2	3	1	3ewma60_223_mean	0.48	60.749	89.298
3	1	1	1	3ewma60_311_mean	0.9748	1.1569	60
3	2	1	1	3ewma60_321_mean	0.48	2.7071	60
3	1	2	1	3ewma60_312_mean	0.9748	3.14	60
3	2	2	1	3ewma60_322_mean	0.48	7.0111	60
3	1	3	1	3ewma60_313_mean	0.9748	34.3157	60

3	2	3	1	3ewma60_323_mean	0.48	60.749	89.400
1	1	1	2	3ewma120_111_mean	0.6097	1.9861	120
1	2	1	2	3ewma120_121_mean	0.2513	3.7216	120
1	1	2	2	3ewma120_112_mean	0.6097	4.8247	120
1	2	2	2	3ewma120_122_mean	0.2513	9.6782	120
1	1	3	2	3ewma120_113_mean	0.6097	48.1767	120
1	2	3	2	3ewma120_123_mean	0.2513	67.023	120
2	1	1	2	3ewma120_211_mean	0.6097	1.9861	120
2	2	1	2	3ewma120_221_mean	0.26	3.722	120
2	1	2	2	3ewma120_212_mean	0.6097	4.8247	120
2	2	2	2	3ewma120_222_mean	0.26	10.1432	120
2	1	3	2	3ewma120_213_mean	0.6097	48.1767	120
2	2	3	2	3ewma120_223_mean	0.26	68.0577	120
3	1	1	2	3ewma120_311_mean	0.5817	1.9903	120
3	2	1	2	3ewma120_321_mean	0.26	3.722	120
3	1	2	2	3ewma120_312_mean	0.5817	4.771	120
3	2	2	2	3ewma120_322_mean	0.26	10.1432	120
3	1	3	2	3ewma120_313_mean	0.5817	46.6159	120
3	2	3	2	3ewma120_323_mean	0.26	68.0577	120

APPENDIX 6B. THE STATISTICAL ANALYSIS OF THE SIMULATION RESULT WITH STANDARD CONTROL LIMITS

ANOVA: False Alarm versus Cyc, Noi, Sig, pre(batch size), M(chart)

```
Factor      Type   Levels  Values
Cyc         fixed       3  1, 2, 3
Noi         fixed       2  1, 2
Sig         fixed       3  1, 2, 3
pre(batch)  fixed       2  1, 2
M(chart)    fixed       3  1, 2, 3

Analysis of Variance for False Alarm

Source           DF       SS      MS       F      P
Cyc               2   0.0038  0.0019    0.09  0.915
Noi               1   7.4030  7.4030  347.02  0.000
Sig               2   0.0000  0.0000    0.00  1.000
pre(batch)        1   3.7093  3.7093  173.88  0.000
M(chart)          2  12.9649  6.4824  303.87  0.000
Cyc*Noi           2   0.0071  0.0036    0.17  0.846
Cyc*Sig           4   0.0000  0.0000    0.00  1.000
Cyc*pre(batch)    2   0.0003  0.0001    0.01  0.993
Cyc*M(chart)      4   0.0001  0.0000    0.00  1.000
Noi*M(chart)      2   0.1508  0.0754    3.53  0.034
Error            85   1.8133  0.0213
Total           107  26.0527

S = 0.146058   R-Sq = 93.04%   R-Sq(adj) = 91.24%
```

ANOVA: Detection Time versus Cyc, Noi, Sig, pre (batch), M (chart)

```
Factor      Type   Levels  Values
Cyc         fixed       3  1, 2, 3
Noi         fixed       2  1, 2
Sig         fixed       3  1, 2, 3
pre(batch)  fixed       2  1, 2
M(chart)    fixed       3  1, 2, 3

Analysis of Variance for Detection Time

Source           DF       SS       MS       F      P
Cyc               2     21.6     10.8    0.36  0.701
Noi               1   1608.3   1608.3   53.07  0.000
Sig               2  45694.4  22847.2  753.88  0.000
pre(batch)        1    443.2    443.2   14.62  0.000
M(chart)          2    318.7    159.4    5.26  0.007
Cyc*Noi           2     42.5     21.3    0.70  0.499
Cyc*Sig           4     43.4     10.8    0.36  0.838
Cyc*pre(batch)    2      1.2      0.6    0.02  0.980
Cyc*M(chart)      4      4.6      1.1    0.04  0.997
Noi*M(chart)      2     34.0     17.0    0.56  0.572
Error            85   2576.0     30.3
Total           107  50788.0

S = 5.50511   R-Sq = 94.93%   R-Sq(adj) = 93.62%
```

CHAPTER 7

CASE STUDY REVISITED

In Chapter 4, we applied regular SPC methods with standard control limits that were expected to produce 370 as ARL_0 for IID normal data. However, in Chapter 6 we found that those control limits do not work on our data as expected, and we searched via simulation for actual control limits and via a robustness study for robust control limits that would yield an ARL_0 approximately equal to 370. In this chapter, we applied the new SPC methods (the modified BM charts and the robust BM charts) incorporating those new control limits to the real datasets from Chapter 4 and compared the performances of various SPC charts on the real datasets.

In Section 7.1, we will compare the performance of regular and modified BM charts with the actual control limits on the real data, and in Section 7.2, we will test the performances of the robust control charts. In Section 7.3, we discuss how to choose between actual and robust control limits for practical applications.

7.1 PERFORMANCES WITH ACTUAL CONTROL LIMITS

Actual control limits for various scenarios have been searched by simulation in Chapter 6. Among those control limits we used the actual control limits found for the simulation model with Cycle 1 and Noise 1 because this simulation model is the closest to the real data set (Pure_sample_data).

For all SPC methods, we preprocessed the raw data to batch means of size 60 to eliminate any cyclic effect. We tested three regular SPC charts on batch means: Shewhart, Tabular Cusum, and EWMA charts. Then we applied each regular SPC chart with modified batch means, the modified BM charts.

Table 7.1 shows the results of the regular batch mean (RBM) charts and the modified batch mean (MBM) charts with actual control limits. There are no false alarms with either the regular BM charts or the modified BM charts. In all cases, the MBM Charts detect the attack earlier than the regular BM charts. All the three regular BM charts detected the attack at the 840^{th} second (= 14 * 60), which is the end of the 14^{th} batch. However, the MBM Shewhart chart was able to detect the attack at the 787^{th} second.

In Figure 7.1 the left graph shows the batch scale and the right graphs show in seconds the scale of the attack batch 14, i.e., the 840^{th} second is the 14^{th} minute. The horizontal lines represent the control limits. The graphs show that the modified charts detect the attack at the beginning of the 14^{th} batch.

Table 7.1 The Number of False Alarms and Detection Time of the SPC charts with Actual Control Limits

	Shewhart (C = 8.349)		Cusum (k=0.5, h =8.82)		EWMA (l =0.2, h = 5.08)	
	MBM	RBM	MBM	RBM	MBM	RBM
False Alarm	0	0	0	0	0	0
Detection Time (second)	787^{th}	840^{th}	789^{th}	840^{th}	790^{th}	840^{th}

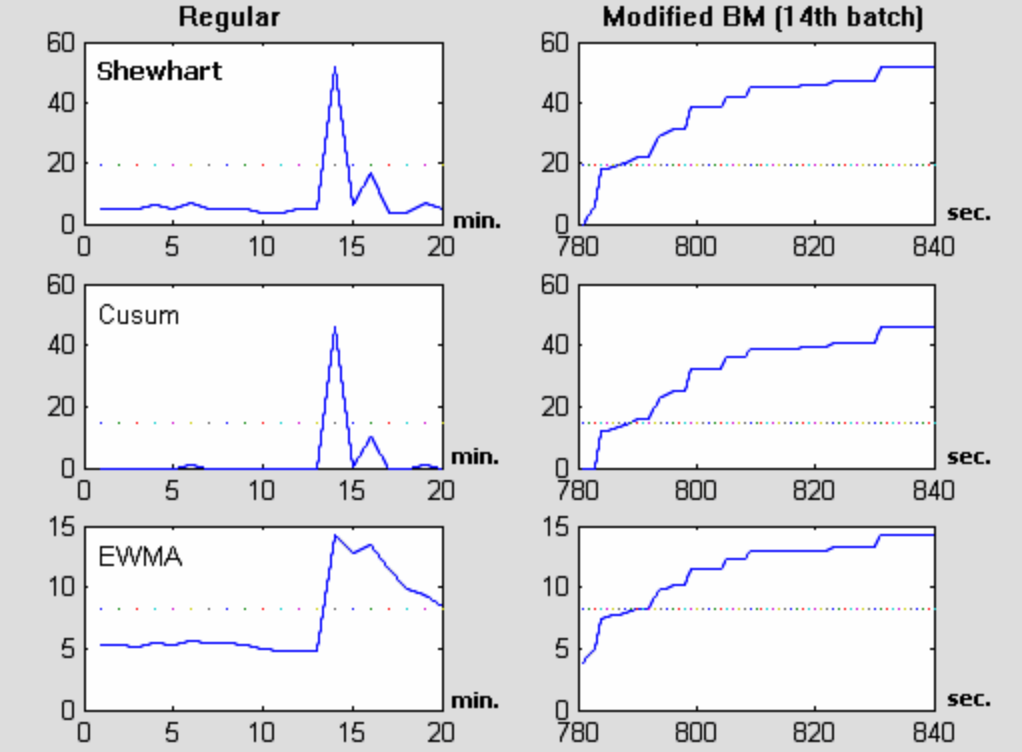


Figure 7.1. Graphical Illustrations of Regular (Left) and Modified (Right) BM Charts with Actual Control Limits

7.2 USING ROBUST CONTROL LIMITS

In this section we repeated the simulation study performed in the previous section, but this time with robust control charts with standard control limits (i.e., robust control limits). Note that actual control limits have to be searched by simulation until they give an actual ARL_0 approximately equal to a target ARL_0, robust control limits do not require simulation and can be calculated analytically or found from standard textbooks.

The performances of the RBM charts and the MBM charts with robust control limits are given in Table 7.2 and Figure 7.2 gives the graphical illustrations of those methods.

Table 7.2. The Number of False Alarms and Detect Time of SPC charts with Robust Control Limits

	Shewhart (B=40, C = 1.237)		Cusum (k=0.05, h =13.47)		EWMA (l =0.01, h = 1.282)	
	MBM	RBM	MBM	RBM	MBM	RBM
False Alarm	0	0	0	0	0	0
Detection Time (second)	N.A.	N.A.	794th	840th	793rd	840th

From Table 7.2, one can note that the MBM charts still detect the attack earlier than the RBM versions of the Cusum and the EWMA charts. Interestingly, neither the MBM nor the RBM Shewhart chart could detect the attack within 1,200 seconds, which is the size of the time span of the Attack_sample_data. It is well known that a large batch size does help to achieve approximate normality and independence, but it can delay the detection of an out-of-control signal. Our results amply demonstrate the problem of having a large batch size.

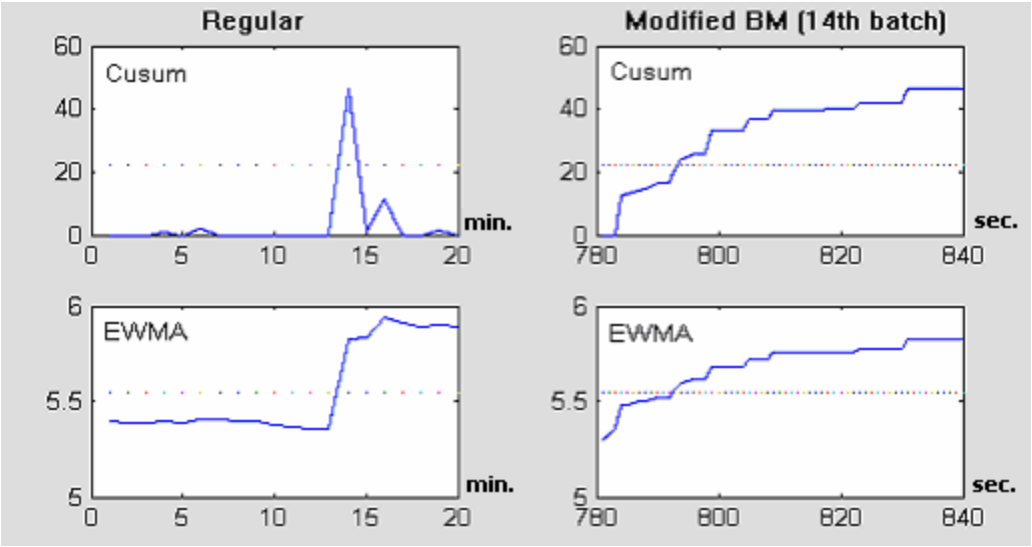


Figure 7.2. Graphical Illustrations of Regular (Left) and Modified (Right) BM Charts with Robust Control Limits

7.3 ACTUAL CONTROL LIMITS VS. ROBUST CONTROL LIMITS

We have seen the advantages of modified SPC charts (MBM charts) over regular SPC charts (RBM charts) in the previous two sections. In this section, we compare the performances of modified SPC charts with the actual and robust control limits.

Table 7.3 summarizes the performance of the MBM charts with actual and robust control limits. One can see that the detection times with robust control limits slower than those with actual control limits. For the MBM Cusum and EWMA charts, the detection times increased from the 789^{th} second to the 794^{th} and from the 790^{th} second to the 793^{rd}, respectively. In the worst case, the MBM Shewhart Chart with robust control limits could not detect the attack at the end of the sample ($1{,}200^{th}$ second).

In the case study, the MBM Shewhart Chart with actual control limits showed the best performance in terms of detection time. However, the difference in detection times is not significant compared with MBM Cusum and EWMA charts that incorporate actual or robust control limits. Moreover, since robust control limits can be determined analytically, it is convenient to use robust control limits. This is unlike actual control limits that require simulation when little information about data properties is available. We do not recommend the modified BM Shewhart chart with robust control limits if an extremely large batch size is involved. If it is possible to determine the actual control limits by simulation or if the required batch size for robust control limits is not that large, we recommend the modified BM Shewhart chart. Otherwise, modified BM Cusum or EWMA charts with robust control limits are the best choice for practical use.

TABLE 7.3.Comparisons of the Performances of the MBM charts with the Actual and the Robust Control Limits

Modified Batch Mean	Control limit	Number of False Alarm	Detection Time in Testing (attack at 784)
Shewhart	Actual c=8.349 B=1 min.	0	787^{th} sec.
	Robust c=1.237 B=40 min.	0	No detection.
Cusum	Actual k=0.5 h=8.82	0	789^{th} sec.
	Robust k=0.05 h=13.47	0	794^{th} sec.
EWMA	Actual l =0.2 L=5.08	0	790^{th} sec.
	Robust l =0.01 L=1.282	0	793^{rd} sec.

CHAPTER 8

CONCLUSIONS

8.1 SUMMARY

In this thesis we applied an SPC monitoring concept to a certain type of traffic data in order to detect a network intrusion. The main assumptions of this study were that an intrusion is a deviation from standard profiles constructed without any intrusion and that the object being monitored is statistically distributed.

We developed a general SPC intrusion detection approach and described it along with the source and the preparation of data used in this thesis. The approach included the source of the data and the preparation of data. From the data, we were able to extract sample data sets for various situations (e.g., idle/busy, attack/no attack), calculate event intensities for each situation and store these sample data sets in the data repository to be used in future research.

Because the sample data had 60-second cycles, the sample data could not be used in a raw state. A regular batch mean (RBM) chart was used to remove the cyclic nature of the sample data. However, the RBM chart monitored the statistic only at the end of the batch, which was unsatisfactory; thus we developed the modified batch mean (MBM) charts that detect the signal or attack faster than regular batch mean chart. Based on the MBM concept, we developed the MBM Shewhart chart, the MBM Cusum chart, and the MBM EWMA chart and studied the performances of these new methods on simulated data.

The MBM Charts can be applied two ways, either by using actual control limits or by using robust control limits. The actual control limits must be determined by simulation, but the robust control limits require only the use of the recommended values. Three robust MBM charts were developed. The robust MBM Shewhart chart was developed on the basis of choosing appropriate values for batch size. The robust MBM Cusum chart and robust MBM EWMA chart were developed on the basis of choosing appropriate values for charting parameters. According to our simulation studies, the robust EWMA chart has the best detection time performance and its control limits can be easily determined. So the robust MBM EWMA chart is the one recommended for use.

8.2 FUTURE RESEARCH

Intrusion detection holds many possibilities for the use of statistical methods, especially for SPC techniques that have been applied successfully to the service and manufacturing areas. We have focused mainly on how to apply the SPC concept to the IDS area by using a certain type of traffic data from BSM data. We can further investigate various data sources (e.g., TCPDUMP or *sendmail*) to see if there exists a better source that contains more information about traffic that can be read by a monitored system. This will require close collaboration between experts in statistics and in computer science.

Currently, the maximum saving in detection time we can get from SPC charts with the MBM concept is only one batch size. This happens because the observed data are non-negative. When a cycle period is long and, therefore, a batch size is large, the savings in detection time can be significant for a large attack signal. However, when the batch size is small because of a short period, the effect will be probably negligible even

for a large signal. We will further investigate the MBM concept to see if it is possible to gain a savings of more than one batch size. It may be possible when the observed data cam be negative. However, the actual control limits need to be refined, which will affect the detection time. More research is needed to investigate the impact of such adjustments.

Finally, it will be interesting to study guidelines for data preprocessing and the determination of control limits for the application of SPC methods to more general cyclic data. In this thesis, we assume that the cycle period and cycle peak are constants. However, these assumptions may not be true for data from other sources. It is possible that the data may have more than one cycle period or that cycle periods or cycle peaks may be random.

REFERENCES

Anderson, J. P. (1980). Computer Security Threat Monitoring and Surveillance. *Technical report*, James P Anderson Co., Fort Washington, Pennsylvania, April 1980.

Nelson, Barry L. and Yamnitsky, Michael (1998). Input modeling tools for complex problems, In *Proceedings of the 30th conference on Winter simulation*, p.105-112, December 13-16, 1998, Washington, D.C., United States.

Bernstein, S. (1912). Démonstration du théorème de Weierstrass fondée sur le calcul des probabilities. *Comm. Soc. Math. Kharkov* **13**, 1-2.

Biba, K. J. (1977). Integrity Constraints for Secure Computer Systems. *Technical Report*, USAF Electronic Systems Division, April 1977.

Bell, D. E. and LaPadula, L.J. (1973). Secure Computer Systems: *Technical Report* M74-244, The MITRE Corp. May 1973.

Borror, C. M., Montgomery, D. C., and Runger, G. C. (1999). Robustness of the EWMA Control Chart to Nonnormality, *Journal of Quality Technology*, Vol. 23.

Denning, Dorothy E. (1982). *Cryptography and Data Security.* Addison-Wesley, Reading, Massachusetts, 1982.

Denning, Dorothy E. (1987). An Intrusion Detection Model. *IEEE Transactions on Software Engineering*, Number 2, page 222, February 1987.

Department of Defense Standard. (1985). *Department of Defense Trusted Computer System Evaluation Criteria*. DOD 5200.28-STD. US Government Printing Office, December 1985.

Garvey, T. D. and Lunt, Teresa F. (1991). Model Based Intrusion Detection. In *Proceedings of the 14th National Computer Security Conference*, pages 372-385, October 1991.

Garfinkel, Simson and Spafford, Gene. (1991). *Practical Unix Security.* Sebastopol, California, 1991.

Glynn, Peter W. and Whitt, W. (1991). Departure from Many Queues in Series. *Annals of Applied Probability*, Vol. 1, 546-572.

Hawkins, D. M. (1993). Cumulative Sum Control Charting: An Underutilized SPC Tool, *Quality Engineering*, Vol. 5.

Heady, R., Luger, G., Maccabe, A. and Servilla, M. (1990). The Architecture of a Network Level Intrusion Detection System. *Technical Report*, CS, Univ. of New Mexico, August 1990.

Hotelling, H. (1947). Multivariate Quality Control, *Techniques of Statistical Analysis*, edited by Eisenhart, Hastay, and Wallis, McGraw-Hill, New York.

Ilgun, Koral. (1992). *USTAT - A Real-time Intrusion Detection System for UNIX*. Master's Thesis, University of California at Santa Barbara, November 1992.

Jiang, Wei. (1999). *Charting Techniques in Integrated APC and SPC Environments*. Ph. D. Thesis, The Hong Kong University of Science & Technology.

Johnson, N. L. (1949). Systems of frequency curves generated by methods of translation, *Biometrika*, 36 (1): 297–304.

Kumar, Sandeep. (1995). *Classification and Detection of Computer Intrusions*. Ph.D. Dissertation, August 1995.

Lampton, B. W. (1974). Protection. *Operating System Review*, Vol. 8, Number 1, January 1974.

Lee, Wenke (1999). *A Data Mining Framework for Construction Features and Models for Intrusion Detection Systems*, Ph.D. Dissertation, Columbia University, New York.

Lee, Wenke and Stolfo, Salvatore J. (1999). *Data Mining Approaches for Intrusion Detection*, Comuter Science Department, Columbia University, New York.

Longley, Dennis and Shain, Michael. (1987). *Data and Computer Security: Dictionary of Standards, Concepts, and Terms*. Stockton Press, New York.

Lunt, Teresa F. (1993). A survey of intrusion detection techniques. In *Computers and Security*, 12(1993), pages 405-418.

Mukherjee, B., Heberlein, L.T. and Levitt, Karl N. (1994). Network Intrusion Detection, *IEEE Network*, May/June 1994, pages 26-41.

Miller, B. P., Koski, D., Lee, C., Maganty, V., Murthy, R., Natarajan, A. and Steidl, J. (1995). *Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services*. Computer Sciences Department, University of Wisconsin, 1995.

Montgomery, D. C. (2001*). Introduction to Statistical Quality Control*, 4th ed., Wiley, New York, NY.

Moskiwitz, H., Plante, R. D., and Wardell, D. G. (1994). Using Run-Length Distributions of Control Charts to Detect False Alarms, *Production and Operations Management*, Vol. 3. No. 3.

Page, E. S. (1961). Cumulative Sum Charts, *Technometrics* 3, page 1-9.

Power, Richard. (1995). *Current and Future Danger*, Computer Security Institute, California.

Russell, Deborah and Gangemi, G. T. Sr. (1991). *Computer Security Basics*. O'Reilly & Associates, Inc., California.

Runger, G. C. and Willemain, T. R. (1995). Model-Free Control of Autocorrelated Processes, *Journal of Quality Technology*.

Runger, G. C. and Willemain, T. R. (1996). Batch Means Control Charts for Autocorrelated Data, *IIE Transactions*, Vol. 28.

Shewhart, W. A. (1931). *Economic Control of Quality of Manufactured Product*. Van Nostrand, N. Y.

Smaha, Steve. (1992). Qeustions about CMAD. In *Proceedings of the Workshop on Future Directions in Computer Misuse and Anomaly Detection*, Davis, California, March 1992.

Smaha, Stephen E. (1988). An Intrusion Detection System. In *Fourth Aerospace Computer Security Applications Conference*, pages 37-44, Tractor Applied Science Inc., Texas, December 1988.

Spafford, Eugene. H. (1989). Crisis and Aftermath. *Communications of the ACM*, 32(6): 678-687, June 1989.

Crosbie, Mark and Spafford, Eugene. H. (1995). Defending a Computer System Using Autonomous Agents. *Technical Report* CSD-TR-95-022, Department of Computer Sciences, Purdue University, 1995.

Spafford, Eugene H. (1996). *Security Seminar*, Department of Computer Sciences, Purdue University, Jan 1996.

Swain, J. J., Venkatraman, S., and Wilson, J. R. (1988). Least squares estimation of distribution functions in Johnson's translation system. *Journal of Statistical Computation and Simulation* 29: 271–297.

Teng, H. S., Chen, K. and Lu, S. C. (1990). Security Audit Trail Analysis Using Inductively Generated Predictive Rules. In *Proceedings of the 11th National Conference on Artificial Intelligence Applications*, pages 24-29, IEEE, IEEE Service Center, Piscataway, NJ, March 1990.

Wagner, M. A. F. and Wilson, J. R. (1996). Using univariate Bezier distributions to model simulation input processes. *IIE Transactions* 28: 699–711.

Woodall, W. H., and Adams, B. M. (1993). The Statistical Design of CUSUM Charts. *Quality Engineering*, 5(4): 559-570.

Ye, N., Li, X., Chen, Q., Emran, S. M., and Xu, M. (2001). Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data, *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, Vol. 31, No. 4, July 2001.

Ye, N., Emran, S. M., Chen, Q., and Vilbert, S. (2002). Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection, *IEEE Transactions on Computers*, Vol. 51, No. 7, July 2002.

Ye, N., Vilbert, S., and Chen, Q. (2003). Computer Intrusion Detection Through EWMA for Autocorrelated and Uncorrelated Data, *IEEE Transactions on Reliability*, Vol. 52, No. 1, March 2003.