DESIGN AND IMPLEMENTATION OF A SECURE AND SEARCHABLE AUDIT LOGGING SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

DAVUT İNCEBACAK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

MAY 2007

Approval of the Graduate School of Informatics

_____

Assoc. Prof. Dr. Nazife BAYKAL

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Yasemin YARDIMCI

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Dr. Kemal BIÇAKCI	Assoc. Prof. Dr. Yasemin YARDIMCI
Co-Supervisor	Supervisor

Examining Committee Members

Assoc. Prof. Dr. Nazife BAYKAL	(METU, II)	_____
Assoc. Prof. Dr. Yasemin YARDIMCI	(METU, II)	_____
Dr. Kemal BIÇAKCI	(METU, II)	_____
Dr. Ali ARİFOĞLU	(METU, II)	_____
Assist. Prof. Dr. A. Aydın SELÇUK	(Bilkent, CENG)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Davut İNCEBACAK

Signature : _____

ABSTRACT

DESIGN AND IMPLEMENTATION OF A SECURE AND SEARCHABLE AUDIT LOGGING SYSTEM

İncebacak, Davut
M.Sc., Department of Information Systems
Supervisor Assoc. Prof. Dr. Yasemin YARDIMCI
Co- Supervisor Dr. Kemal BIÇAKCI

May 2007, 117 pages

Logs are append-only time-stamped records to represent events in computers or network devices. Today, in many real-world networking applications, logging is a central service however it is a big challenge to satisfy the conflicting requirements when the security of log records is of concern. On one hand, being kept on mostly untrusted hosts, the logs should be preserved against unauthorized modifications and privacy breaches. On the other, serving as the primary evidence for digital crimes, logs are often needed for analysis by investigators.

In this thesis, motivated by these requirements we define a model which integrates forward integrity techniques with search capabilities of encrypted logs. We also implement this model with advanced cryptographic primitives such as Identity

Based Encryption. Our model, in one side, provides secure delegation of search capabilities to authorized users while protecting information privacy, on the other, these search capabilities set boundaries of a user's search operation. By this way user can not access logs which are not related with his case. Also, in this dissertation, we propose an improvement to Schneier and Kelsey's idea of forward integrity mechanism.

Keywords: Forward Integrity, Audit Log, Identity Based Encryption, Logging Systems, Applied Cryptography

ÖZ

GÜVENLİ VE ARANABİLEN KAYIT TUTMA SİSTEMİ TASARIMI VE UYGULANMASI

İncebacak, Davut
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Danışmanı: Doç. Dr. Yasemin YARDIMCI
Yardımcı Tez Danışmanı: Dr. Kemal BIÇAKCI

Mayıs 2007, 117 sayfa

Kayıtlar bilgisayar ve ağ cihazlarında gerçekleşen olayları gösteren içinde ne zaman oluşturulduğu bilgisi bulunan sadece oluşturulduğu dosyanın sonuna eklenebilen günlüklerdir. Bir çok gerçek ağ uygulamasında kayıt tutmak merkezi hizmetlerden biridir bununla beraber kayıtların güvenliği söz konusu olduğunda çelişen gereksinimleri sağlamak çok zordur. Bir tarafta, kayıtlar güvenilir olmayan makinalarda tutulduğu için izinsiz modifikasyonlara ve gizlilik ihlallerine karşı korunmalı diğer taraftan dijital suçlarda birincil delil olması nedeniyle araştırmacılar tarafından kayıtların sıkça analiz edilmesi sağlanmalı.

Bu tezde, yukarıdaki gereksinimleri göz önüne alarak, ileri bütünlük mekanizması ile şifrelenmiş kayıtların arama yeteneği birleştirilerek bir model tanımladık. Ayrıca bu modelin Kimlik Tabanlı Şifreleme gibi modern kripto araçları ile uygulamasını gerçekleştirdik. Tanımladığımız model, bir taraftan bilgi gizliliğini koruyarak arama yeteneklerinin yetkili kişilere delegasyonunu sağlar diğer taraftan bu arama yetenekleri kullanıcının arama işlemini sınırlar. Bu şekilde kullanıcı kendi problemi ile alakalı olmayan kayıtlara erişememiş olur. Ayrıca bu tezde, Schneier ve Kelsey nin ileri bütünlük mekanizmasına bir geliştirme önerdik.

Anahtar Kelimeler: İleri Bütünlük, Denetlenebilir Kayıtlar, Kimlik Tabanlı Şifreleme, Kayıt Sistemleri, Uygulamalı Kriptografi

This thesis is dedicated to:

My Lovely Family

For their endless support,
For their love…

ACKNOWLEDGEMENT

It is a pleasure for me to express my sincere gratitude to Dr. Kemal Bıçakcı for his patience, encouragement and guidance throughout the study. I greatly appreciate his helps when I take my first steps in Academic Writing and always being with me in every phase of this Thesis.

I would like to also express my special gratitude to my supervisor Assoc. Prof. Dr. Yasemin Yardımcı for her support, guidance, helps and suggestions throughout my research.

I am very grateful to Yusuf Uzunay for the reviewing of my thesis and continuous support.

I appreciate Ayşe Ceylan, Sibel Gülnar and Ali Kantar in the institute for their kindness since the beginning of my M.Sc. study.

Finally, I will also never forget the unending support my family has provided me with during all the hard times.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

AES	: Advanced Encryption Standard
ASCII	: American Standard Code for Information Interchange
CA	: Certificate Authority
CERT	: Computer Emergency Response Team
CR	: Certificate Repository
CRL	: Certificate Revocation List
DES	: Data Encryption Standard
DoS	: Denial of Service
HMAC	: Keyed-hash message authentication code
HTTP	: Hyper Text Transfer Protocol
IBE	: Identity Based Encryption
ID	: Identity
IDS	: Intrusion Detection Systems
I-SMS	: Intelligent Security Management System
JCA	: Java Cryptographic Architecture
MAC	: Message Authentication Code
PKG	: Private Key Generator
PKI	: Public Key Infrastructure
RA	: Registration Authority
RC4	: Rivest Code 4
RC5	: Rivest Code 5
RSK	: Random Symmetric Key
UML	: Unified Modeling Language

CHAPTER I

INTRODUCTION

Logs are append-only, time stamped records which are designed to represent some events that occurred in computers or network devices. Originally, logs were used primarily for troubleshooting problems in computer system [1], but logs now constitute a very important part of maintaining the security of network, since logs can assist an investigator or administrator from debugging system or network problems to providing useful data for investigating malicious activity. Therefore, as their analysis can reveal whether an intrusion occurred or not, log files are lucrative targets for attack.

Administrators of network systems may face with lots of problems foremost of which is the issue of proving identity of an offender. For example, users might try to have unauthorized access to resources by impersonation, insult or threat using fake email addresses. In these cases, the problem of proving the identity of the offender is not that easy. Since the suspected user may simply say that his password was stolen, all the traces must be analyzed to find out who is the real offender. Most useful traces left behind are the logs and the only way to prove the identity of an offender is to analyze various proxy servers' logs, internal server logs, and web logs and try to find out entries which matches with that particular user. To identify the source of a crime committed using a computer, there is a need

for digital evidence and most of the evidence is extracted from these logs. In other words, logs are the key to solve a criminal case [2, 3].

Besides using log files as digital evidence, there are other usage areas of logs. One of them is in intrusion prevention and detection systems. In today's world, it is almost impossible to see a company that is not connected to the Internet. As the rate of internet connectivity increases, a corresponding increase in the number of attacks against these companies and to their networks is observed. This fact of dramatic growth in reported incidents of security breach over past years is demonstrated in the reports generated from the Computer Emergency Response Team Coordination Center (CERT Coordination Center) databases [4].

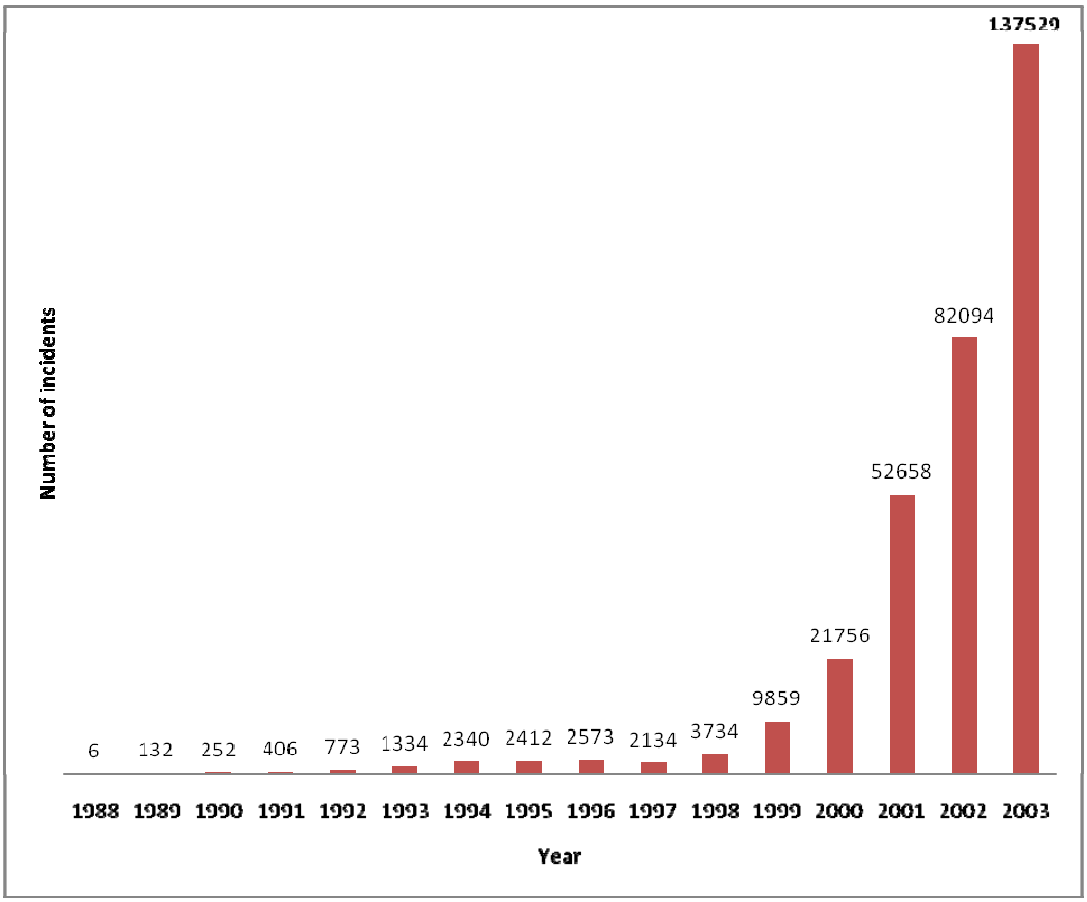


Figure 1 : Number of Incidents

After 2003, as it can be seen above figure, CERT Coordination Center no longer published the number of incidents reported because attacks against Internet-connected systems have become so commonplace. Only a single successful attack to a company network may cause the company to lose a lot of respect and profit. if the primary business activity highly depends on networking systems, even the company might become bankrupt. That is the reason why many research efforts concentrate on preventing and detecting such attacks. Consequently, various intrusion prevention systems and intrusion detection systems were developed [5]. Logs are not only the most important part of these systems as a feedback but also logs from different sources constitute the base for the intrusion prevention and detection mechanisms. The main reason is that log files include the data necessary for intrusion prevention and detection systems detecting patterns of misuse of system resources.

Although, Intrusion Detection Systems (IDS) provide a good solution to meet the security requirements of companies, blindly trusting to those systems can mislead companies. In order to provide accurate and sufficient information, IDS should improve its intelligence which usually requires a trusted audit log database. In addition to this, since IDS generally works on network layer, there is a need to monitor potential threats at the application layer. Here, monitoring logs again helps; application logs are one step towards a solution of this problem [6].

Log files are also used by system administrators to decide on the efficiency of the network and can help solving network problems. Decision support, increasing network performance and improving availability are all possible by careful examination of system log files.

Another usage of logs is tamper detection of copyrighted materials such as documents, music, video, and software. Software tamper resistance is the protection of software codes against reverse engineering. Most of the time, since software works under the control of attacker, it is hard to provide full tamper resistance. However, software tamper detection can be achieved by checking the

integrity of software and logging in a way that hackers can not forge or erase. Failed integrity check in the logged data indicates that there is a tamper on the software. This information can be used for tamper resistance of software such as software of music providers which requires connecting to distributors of this software to be able to gain access to service. When the hacker connects to the software provider, software provider controls the logged data and if there is failed integrity check in the logged data this shows ongoing tampering process [7].

As it can be understood from the discussion above, logs are the main source of information for digital security hence security and reliability of information provided by these logs has utmost importance.

I.1 Log Protection

There are many methods to protect logs. Protection methods of logs can be classified in two parts: external protection of logs, internal protection of logs.

I.1.1 External Protection of Logs:

In this kind of log protection, security of logs is provided by the underlying system in which the logs were produced [8, 9]. Techniques used to defend logs are:

1- **Making Log Files Append Only:**

The only operation possible on append-only log files is to append data to its previous content but not change the content itself. By this way, attacker can not modify log file as he wishes.

2- Set Proper Permissions:

Log entries are mostly appended to log file in an ASCII standard format, which is a universally understood character set for common devices and applications. Allowing read access to log files reveals valuable information to attacker. Therefore, permissions on log files should be configured so that only the administrator or delegated user is able to access.

3- Password Protection:

Together with other techniques, using password to protect log files or the directory that includes log files allows controlled access to the log entries.

4- Create Duplicate Log Files:

Log files can be written to more than one location in the system. By this way when an attacker comprises one set of log files and makes modification on this file, the other set will reveal the presence of the attacker.

5- Hide Log Files:

Mostly logs are stored in a default location which is publicly known. Once an attacker gains access to the system, he tries to modify logs to cover up his tracks and to avoid detection by system, network, and security administrators. Hence, that default location will be checked to reach the logs by the attacker. By keeping logs in a different part in the system rather than in default location and, if possible, not giving obvious names such as ".log", "xxxlogxx.dat" will provide a means of protection to log files.

Rather than using each technique on its own, using together provides better protection.

I.1.2 Internal Protection of Logs:

Internal mechanisms mean using cryptographic techniques to protect log entries from attacks. For instance, encryption of logs is the straightforward solution to protect the confidentiality of log entries. Also, there are cryptographic techniques which guarantee the integrity of log entries.

I.2 Scope of Thesis

In this thesis, our aim is to provide robust logging mechanism in an untrusted environment. Logging is ubiquitous, almost all systems produce logs and most of the time log files are kept on local machine. If the security of the logs is not provided internally, security of the larger system defines the security of the logs. System dependent mechanisms may prevent processes from illicitly accessing log information. However, logs protected only by these mechanisms can be read when mechanism are failed or bypassed. They can protect the secrecy of data as cryptography, but if they fail or are evaded, the data becomes visible. A sophisticated attacker who has gained control of the logging machine can read, modify log files as he wish. Therefore, we have constructed secure audit logs of which security features are internal. By this way, since cryptographic keys are not stored in the system, even the security of the system is surpassed by an attacker, log files can not be modified without detection.

Detection of any modification can be achieved by integrity checking with the usage of some cryptographic tools but it is not enough for the protection of the logs. Sometimes, main aim of the attacker can only be to read content of the logs

because log files are like a treasure map; if they are dug carefully, valuable information can be gathered. For example, vulnerabilities of the system can be found out from the log files and by using this information, attacker can exploit the system. Also some log files may include personal information such as credit card number, access pattern of a user. So, readability of log file may harm user privacy. Beside, logs reveal presence of attackers. Since deleting whole log files, shows that an intrusion occurred in the system, preference of attacker is deleting log records that reveal their presence but to do this, they have to read the log file. We provide a robust logging mechanism by encrypting log entries using random symmetric key in a way that prevent unauthorized access to sensitive data contained in the log messages.

Encryption of log entries makes hard to retrieve data selectively. Therefore, we define a model for searching on our robust logging mechanism and implement it using Identity Based Encryption (IBE) and other encryption techniques on IPFilter Firewall logs. In our mechanism, log entries and keywords extracted from these log entries are encrypted in a way that allows the investigator to determine which log entries contain a certain keyword or keyword set. By this way, an investigator could analyze the logs effectively but can not get information unrelated with his case.

I.3 Outline of Thesis

This dissertation is composed of 6 chapters. In this chapter, we first define the importance of logs by giving real world examples. After the importance of protecting logs is emphasized, we finally present scope and outline of this thesis.

We have dedicated chapter 2 for background information. Since, our study is based on cryptographic techniques of which includes relatively new encryption technique called Identity-based encryption, we in this chapter, present an overview of the basic concepts of cryptography and some security issues.

In chapter 3, we give an overview of previous studies which are on secure audit logging and searching on encrypted data. In the last part of this chapter, we analyze previous studies and discuss their shortcomings.

In chapter 4, we first define the problem and describe the design requirements of secure audit logs and how these requirements can be applied is explained. Then we give our solution on secure audit logging mechanism and compare our scheme with other proposed mechanisms.

In chapter 5, detailed explanation of implementation of our solution on IPFilter Firewall logs are given.

Finally we wrap up our thesis with a conclusion and future work in Chapter 6.

CHAPTER II

BACKGROUND

The work described in this thesis is based on cryptographic techniques and some security related issues. Before going into more detail about our work, in this chapter, we present an overview of the basic concepts of cryptography and computer security

II.1 Computer Security

"To be free from harm" is perhaps the closest short description of secure. Anything valuable that can be misused may need some type of protection such as audit logs. Security which may refer to any measures taken to protect something manifests itself in many ways in accord with the situation and requirement [10]. Examples of security in the real world include locks on doors, alarms in our cars, police officers. In computerized world, security rests on confidentiality, integrity and availability.

II.1.1 Confidentiality

Confidentiality is concealment of sensitive information from unauthorized persons and sensitive facilities from physical, technical or electronic penetration or

exploitation. The need for keeping information secret arises from using computer in sensitive fields. [11].

II.1.2 Integrity

Integrity may be defined as the condition existing when data is unchanged from its source and has not been accidentally or maliciously modified, altered, or destroyed. Integrity is classified in two categories: data integrity (the content of the information) and origin integrity (the source of the data, often called authentication). Dealing with integrity is harder than dealing with confidentiality since it relies on assumptions about the source of the data and about trust in that source – two keystones of security that are often overlooked.

II.1.3 Availability

Availability refers to the ability to obtain or access information or resource when necessary. The increasing dependence of the state on networked applications, the Internet, Intranets, etc., require the creation of an environment to provide a high level of data availability. As a general rule, the more critical component is, the higher its availability will be. The aspect of availability that is relevant to security is that someone may deliberately prevent accessing services or data by making it unavailable. This kind of security incidents is called denial of service attacks which can be the most difficult to detect, an example of this kind of attack to RSA security company which is one of the biggest security firm in the world [12].

II.2 Cryptography

"There are two kinds of cryptography in this world: cryptography that will stop your kid sister from reading your files, and cryptography that will stop major governments from reading your files. This book is about the latter."

--Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C.

As Bruce Schneier said this part is about the latter. The term *cryptography* is derived from the Greek words "kryptos," standing for "hidden," and "gráphein," standing for "write." accordingly, the meaning of the term *cryptography* is best paraphrased as "hidden writing." According to Request for Comments (RFC) 2828 [13], cryptography refers to the "mathematical science that deals with transforming data to render its meaning unintelligible (i. e., to hide its semantic content), prevent its undetected alteration, or prevent its unauthorized use. If the transformation is reversible, cryptography also deals with restoring encrypted data to intelligible form." In brief, Cryptography can be defined as science of writing in secret but cryptography not only protects data unauthorized reading or alteration, it can also be used for user authentication that is the process of proving one's identity. There are, in general, three types of cryptographic schemes: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below.

Before moving on detail about these issues, let's look at some *cryptography* jargon [14]. The data that is wanted to keep secret is referred to as plaintext (some call it cleartext). Plaintext can be human readable text file, such as a memo or it could be a binary file which makes no sense to human but it is understood by computers perfectly. The operation of converting plaintext to gibberish is called encryption and gibberish product of encryption operation is referred as ciphertext. The operation that converts ciphertext back to plaintext is called decryption.

For the encryption and decryption operation, an algorithm is used. In computer cryptography, many encryption algorithms, also named as cipher, exist and they sometimes include complex mathematical operations or simple bit manipulations.

To encrypt plaintext into ciphertext, cryptographic algorithms use key. It works in the same way that conventional key works. To protect a house, a door is built but it is not enough for security. One more thing is need to be enabling access to only living family of that house – a lock on the door. Lock only operates by inserting and turning a key. Tumblers and mechanism of lock operates with the key in a prescribed way to activate a barrier that prevents the door from being opened. Again, key is used to unlock door. In cryptography, to protect sensitive information, an algorithm is installed (a lock on the door) that enables encrypting and decrypting data on the computer (the door). To operate algorithm, secret number is entered as an input (key of lock) and execute it (turning a key). The cryptographic algorithm performs its steps and converts plaintext into ciphertext.

Last but not least, in cryptography, one basic principle is that the security of a cryptographic algorithm should depend on only the secrecy of the key not the secrecy of the encryption algorithm because attackers can deduce algorithm with or without help from the founder of algorithm. Never in the history of cryptography has someone been able to keep an algorithm secret. This is the reason why open source cryptographic algorithms are very popular today and used in very important Internet protocols [14].

II.2.1 Symmetric Cryptography

Symmetric or secret key cryptography has been in use for thousands of years and contains any type of cryptographic algorithm in which the same key is used both to encrypt and to decrypt the same shared secret. Since the single key is used for both encryption and decryption, it is critical that this key is kept strictly private. Otherwise, an intruder can easily encrypt and decrypt messages at will. Due to this

reason, symmetric ciphers are often referred to as private key, secret key, or shared key ciphers. Well known examples of this algorithm are DES, AES, 3-DES, RC4, RC5 and etc.

Mathematically, symmetric encryption and decryption (Figure 1) can be represented by the following, where **E** is an encryption function, **D** is a decryption function, **K** is the shared key, **M** is a plaintext message, and **C** is the corresponding ciphertext message:

Encryption: $C = E_K (M)$

Decryption: $M = D_K (C)$

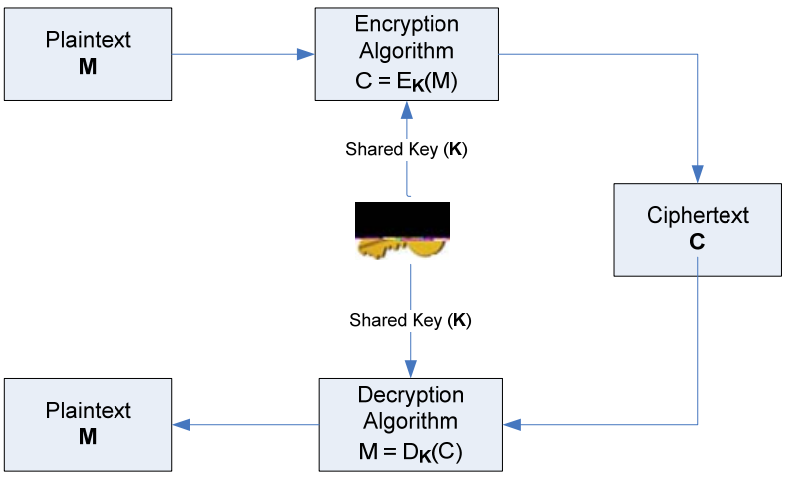


Figure 2 : Symmetric encryption

Symmetric encryption algorithms are typically designed to minimize the computation required to encrypt or decrypt data and therefore they operate at high speeds.

Symmetric encryption is much faster than asymmetric encryption, but main problem with symmetric cryptographic algorithms is that separate keys must be stored for each communicating pair. This means that the number of keys that need

to be maintained as secrets by all communicating parties grows rapidly as the number of parties increase. If you want to communicate with N parties using symmetric encryption, since each communication requires a symmetric secret key, the total number of keys required equals N*(N-1)/2 [15].

In addition to the symmetric key proliferation problem; a second problem about key exchange results from the fact that communicating parties must one way or another share a secret key before any secure communication can be initiated, and both parties must then ensure that the key remains secret.

II.2.2 Asymmetric Cryptography

Asymmetric Cryptography also named as Public Key Cryptography is first proposed by Diffie and Hellman in 1976. In contrary to the symmetric systems, in asymmetric cryptography, the keys used in encryption and decryption operations are different from each other. Each entity thus has two keys. Two keys, based on mathematical functions instead of basic bit operations, are correlated in such a way that plain text encrypted with the one key can only be decrypted with the other. Each entity keeps one key secret named as private key and publishes the other one named as public key. If A wants to send a message to B, A just encrypts it with B's public key. Given that B is the only one who has access to the private key, B is the only one who can decrypt the message and access the content. Well known examples of this algorithm are RSA and El-Gamal.

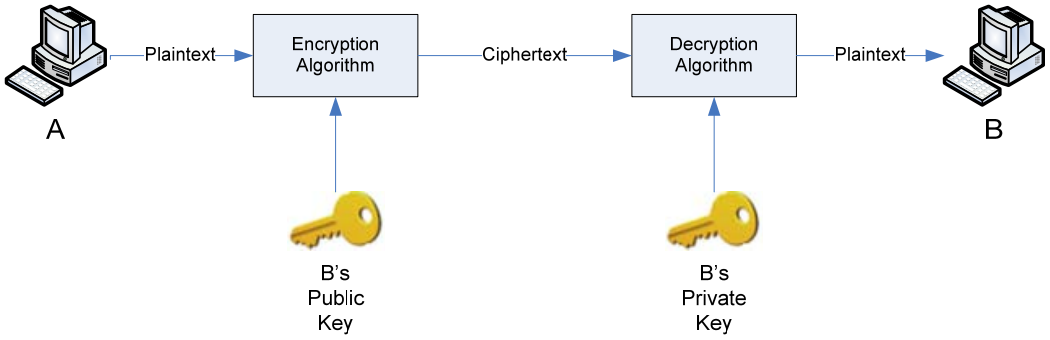


Figure 3 : Asymmetric encryption

By using asymmetric encryption, it is succeeded that only the private key must be kept secret and that private key needs to be kept only by one entity. When it is compared to symmetric encryption, it is an extremely important development for a large environment like internet where most of the time two entities have no previous contact. Although, private key is kept secret by holder, authenticity of the corresponding public key must be guaranteed somehow by a trusted third party, this is mostly succeeded by Certificate Authority (CA) which publishes a public key certificate for an entity stating that the CA testifies that the public key contained in the certificate really belongs to the person, organization, or other entity noted in the certificate. Since private key is kept by only one entity, asymmetric schemes can be used to implement digital signature schemes that enable nonrepudiation. Also, as each entity has one key pair, the total number of required keys for secure communication is much smaller than in the symmetric case.

II.2.3 Digital Certificates

One of the underpinning of the asymmetric techniques is digital certificates. In order to make an asymmetric algorithm such as RSA work, a way is needed to distribute the public key. Digital certificates provide a data structure that binds a public key to an entity in an authentic way. A trusted authority which is trusted by communicating entities signs this data structure. By this way, digital certificates

provide a more complete security solution for the distribution of public key, assuring the identity of all parties involved in a communication. Semi-formally, data structure of a digital certificate may be illustrated like [16]:

```
certificate::=
   {
    issuer name;
    issuer information;
    subject name;
    subject information;
    validity period;
   }
issuer information ::=
   {
    issuer public key;
    signature algorithm identifier;
    hash function identifier
   }
subject information ::=
   {
    subject public key;
    public key algorithm identifier
   }
validity period ::=
   {
    start date;
    finish date
   }
```

II.2.4 Public Key Infrastructure (PKI)

RFC 2822 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

Key elements of the PKI are as follows [17]:

End Entity: Denotes user, devices or any other entity that uses asymmetric cryptography.

Certification Authority (CA): Trusted third party which issues certificates

Registration Authority (RA): helps certification authority on a number of administrative functions.

Certificate Repository (CR): A generic term used to denote any method for storing certificates.

Certificate Revocation List (CRL): used to publish revoked certificates.

The main steps of the certificate management by PKI (Figure 3) are as follows:

1. ***Registration:***

   To be able to benefit from services of PKI, a user first makes itself known to a CA through an RA. Registration begins the process of enrolling in a PKI and continues with generation of key pair; if end entity generates key pair, he passes the public key to the CA. If not, CA generates key pair and passes private key to end entity securely.

2. ***Certification*:**

CA issues and distributes a certificate for the public key of end entity and to a certificate repository

3. Certificate Update:

All certificates have an expiration date. If end-entity wants to renew its certificate before expire, end entity makes a request and CA updates certificate according to the request of the end-entity.

4. ***Certificate revocation*:**

CA revokes certificates in the case of an abnormal situation requiring certificate revocation, for example, compromise of private key, change in affiliation, or name change.

5. ***Certificate retrieval*:**

Certificates of end-entities are retrieved from a certificate repository by end-entities or they may exchange certificates.

6. ***Certificate validation*:**

To validate certificates whether public key is authentic or still in the usage, the end-entities control CRLs from the CRL repository.

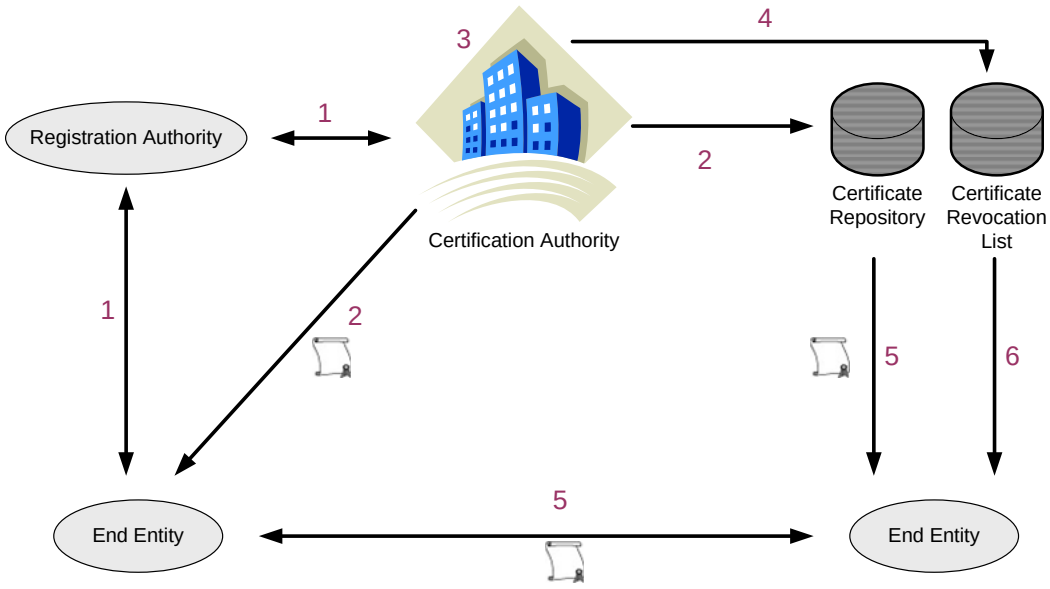


Figure 4 : PKI

II.2.5 IDENTITY BASED ENCRYPTION

Identity-based encryption is a relatively new encryption technique and similar to ordinary public-key systems, involving a private and a public key pair, however a public/private key pair can be produced from any string. Therefore, any publicly available information such as e- mail address, phone number that is uniquely associated with the user identity can be used as to create a public/private key pair for that user [18].

II.2.5.1 Brief History of Identity-Based Encryption

The classic and most widely used schemes in the past for encryption and decryption were the symmetric algorithms. Since one key was used for encryption and decryption, key management problem where secret key between two communicating entity came from was obvious. By the invention of asymmetric encryption schemes in 1977, key management problem was solved to some extent. The new solution, however, gave rise to public key management problem: When

entity A wants to communicate with entity B in a secure way, entity A must the get public key of entity B and entity A has to be convinced that this public key really belongs to entity B not public key of somebody else. In other words, the authenticity of the public key is very important, since it guarantees that encrypted message can only be decrypted by the owner of the corresponding private key. To solve public key management problem, certificates were invented. In a sense, a third party attests that this public key really belongs to entity B.

In 1984, one of the co-founders of RSA algorithm, Adi Shamir, proposed an idea that identity of people e-mail address, phone number can be used as public key. This is a great idea since it eliminated the need for certificates but implementation of this idea was not possible because, at that time, there are only two types of public key schemes namely RSA and El-Gamal encryption. Fundamentally, there is no way a taking an identity of a user and mapping it to the RSA public key or El-Gamal public key. The Idea of Shamir constructing an identity-based encryption (IBE) scheme was left as an open problem. Since then, there were numerous attempts to realize Shamir's idea of identity-based encryption, such as those in [19, 20, 21, 22, 23, 24]. However, none of these proposals were fully satisfactory. Either they did not provide adequate security in their scheme or they were not feasible to implement in practical environments.

In 1999, Dan Boneh and Matthew Franklin started to work on to solve revocation problem of certificate in the existing public key infrastructure. In fact, revocation is hardest problem for certificate management. Once a certificate is issued for a public key, it is hard to revoke that certificate. When private key is lost or stolen, the certificate associated with this private key is no longer valid, corresponding public key in the certificate is no longer valid too. That is why certificate revocation list, certification trees were invented. All the complex technology in the PKI is just to solve revocation of certificates. Their studies showed that Identity Based Encryption would give a very simple solution for the revocation mechanism. If they enable people encrypt messages with IBE using identity of them concatenated with some temporary information such as date of today, this

means every day your public key is changing. For example, on Monday, entity A sends email to entity B using identity of B concatenated with date of Monday, on Tuesday, entity A sends email to entity B using identity of B concatenated with date of Tuesday, so on. Every day your public key changes which means your private key changes once a day, once a week and so on. It is a matter of policy, which information will be concatenated with your identity. When an entity is revoked, key server stops issuing private key for that entity and that entity is revoked [25].

Only in the early 2000's did the emergence of cryptographic schemes based on pairings on elliptic curves result in the construction of a feasible and secure IBE scheme [26]. Boneh and Franklin [27] then presented the first practical and secure IBE scheme based on the Weil pairings.

II.2.5.2 The IBE Algorithm

An IBE scheme uses a trusted third party called a Private Key Generator (PKG) and is based on special type of function called as bilinear map. A bilinear map is defined as a pairing that has specific mathematical properties. To construct an IBE system, it is necessary to find a bilinear mapping algorithm which is *secure*, *computable*, and *efficient*. For the time being, the Weil and Tate pairings are the only recognized algorithms based on elliptic curves to build secure bilinear maps.

Identity Based Encryption scheme consists of four algorithms:

1. **Setup:** $k \in Z^+$, takes a security parameter k and generates *params* (system parameters) and *master key*. *params* will be publicly known and contain description of both finite message space *M* and ciphertext space *C* but *master-key* is only known by the Private Key Generator (PKG). Since, master key is only known by PKG, nobody except PKG is able to construct the Private Key.

2. **Private Key Extraction:** Private key is generated by the PKG. PKG uses the *master-key, params* to generate the private key corresponding to an arbitrary string *ID* such as e-mail address, phone number which will be used as a public key.

3. **Encryption:** Uses *ID* as a public key and *params*, the sender encrypts plaintext message *M* and obtains a ciphertext *C*.

   C = Encrypt (*params*, *ID*, *M*)

4. **Decryption:** Takes as input ciphertext *C, params and private key* which is generated by the Private Key Extraction algorithm and returns plaintext message *M*.

   M = Decrypt (*params*, *private key*, *C*)

Although the actual implementation of IBE is very sophisticated and requires additional information to understand underlying mathematics, to provide better understanding of IBE, a simplified implementation is described below [27]:

PKG initializes the IBE algorithm by picking an elliptic curve, a secret ***s*** (*master-key*) and a point ***P*** on the elliptic curve by the help of random number generator. Then constructs the system parameters (*params)*; ***P*** and ***s*** • ***P***. The operator "•" is a special type of multiplication that multiplies integers with points on elliptic curve. The security of IBE encryption is similar to the other asymmetric algorithms; given ***P*** and ***s*** • ***P***, it should nearly be impossible to compute ***s***. Next, ***P*** and ***s*** • ***P*** are distributed to all users. Mostly, certificate server is used for distribution.

If an entity A wants to send a message in an encrypted way to entity B using IBE, he only needs to know the identity of entity B. Firstly, entity A calculates message digest of entity B's identity (identity of entity B might, for example, be the string

entityB@ii.metu.edu.tr) and maps this message digest to a point $\mathbf{(ID_B)}$ on the elliptic curve. Entity A then calculates a key ***(k)*** by picking a random ***r***:

$$\boldsymbol{k} = \mathbf{Pair}(\boldsymbol{r} \bullet \mathbf{ID_B} , \boldsymbol{s} \bullet \boldsymbol{P})$$

After creating ***k***, entity A encrypts plaintext message (***M)*** and send this encrypted message (**C*)*** together with product $\boldsymbol{r} \bullet \boldsymbol{P}$ to entity B.

$$\boldsymbol{C} = \mathbf{E}_{\boldsymbol{k}}\ \boldsymbol{(M)}$$

When entity B receives ***C,*** he needs corresponding private key for ***k.*** If he has not yet acquired a private key, since IBE private keys are only issued by PKG, he has to authenticate himself to the PKG. After authentication process is completed, PKG calculates message digest of entity B's identity and maps message digest to a point $\mathbf{(ID_B)}$ on the elliptic curve. Then, PKG calculates $\boldsymbol{s} \bullet \mathbf{ID_B}$ (private key) and returns it to him.

Now, entity B has both $\boldsymbol{s} \bullet \mathbf{ID_B}$ and encrypted message ***C***. Entity B can recover the key ***k*** by using $\boldsymbol{s} \bullet \mathbf{ID_B}$.

$$\boldsymbol{k} = \mathbf{Pair}(\boldsymbol{s} \bullet \mathbf{ID_B} , \boldsymbol{r} \bullet \boldsymbol{P})$$

As entity B is the only person who knows his private key, $\boldsymbol{s} \bullet \mathbf{ID_B}$, no one else can calculate ***k***. After creating ***k***, entity B decrypts cipher text (***C)*** and gets the plaintext message M.

$$\boldsymbol{M} = \mathbf{D}_{\boldsymbol{k}}\ \boldsymbol{(C)}$$

How IBE works in practice is depicted below:

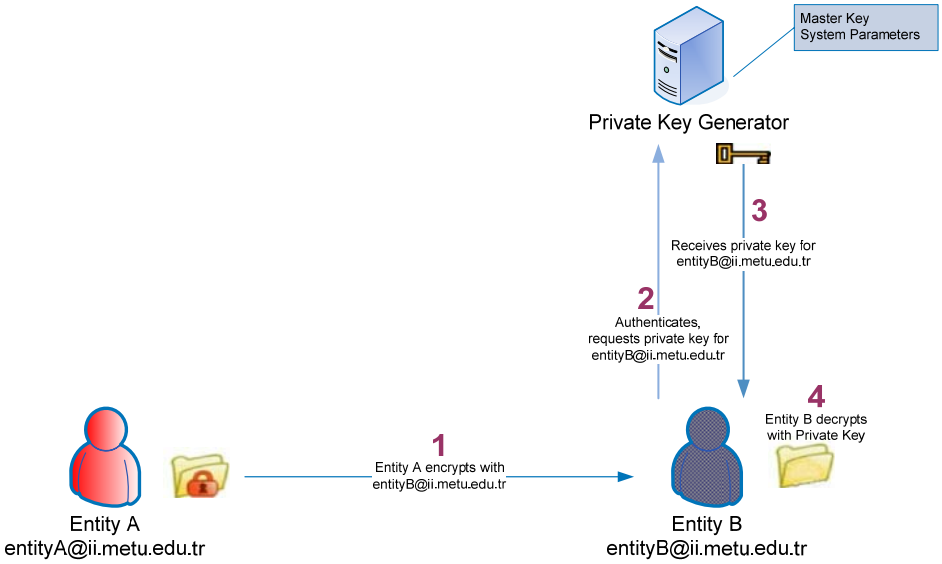


Figure 5 : IBE Online Version

Identity-based encryption's advantages make sense for offline systems. Entity A does not need to communicate with a trusted third party to get public key of entity B and after, entity B receives private key for his identity, he does not need to communicate with PKG. Therefore, IBE can work offline.

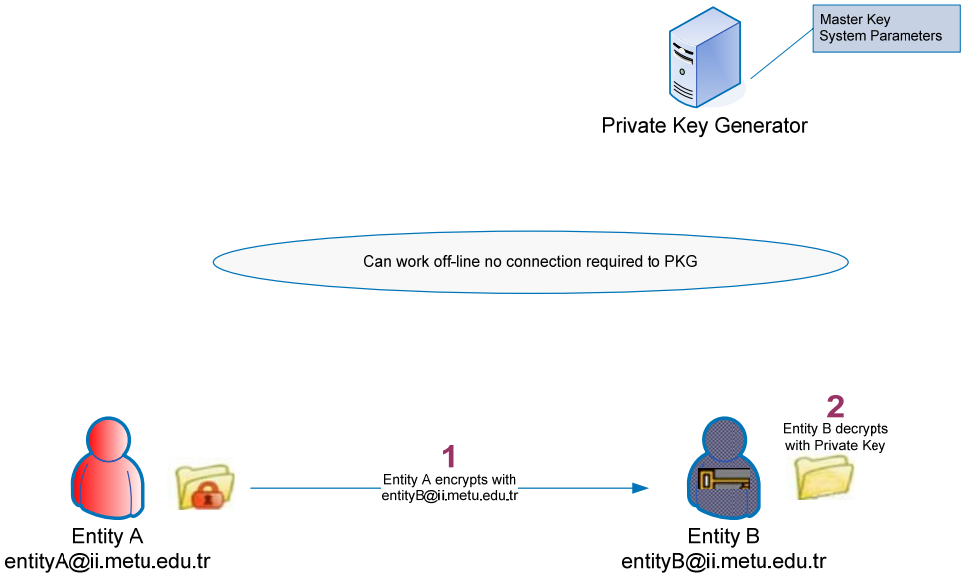


Figure 6 : IBE Offline Version

II.2.6 One Way Hash Functions

A one-way hash or message digest function is a mathematical function that takes an arbitrary-length input and produces a fixed-length output (hash). A small change in the input of hash function can cause the hash value to change intensely. For example, if one bit is flipped in the input, because of **avalanche effect,** on average half of the bits in the hash value will flip as a result [28].

Hash function holds following properties, where **M** is an input message and **H (M)** fixed-length output (hash):

- **One-way:** Given a hash H (M), it is difficult to find the message M.

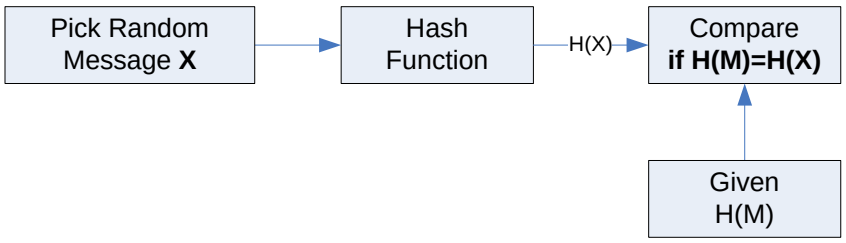


Figure 7 : One-way property of Hash Function

- **Second pre-image resistant:** Given a message M_1, it is difficult to find another message M_2 such that H (M_1) = H (M_2).

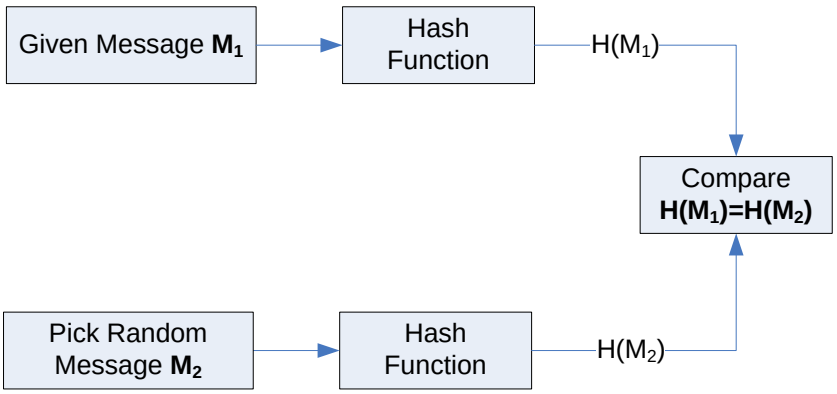


Figure 8 : Second pre-image resistant property of Hash Function

- **Collision resistant:** A hash function is collision resistant if it is difficult to find two messages that hash to the same output: M_1 and M_2 such that H (M_1) = H (M_2).

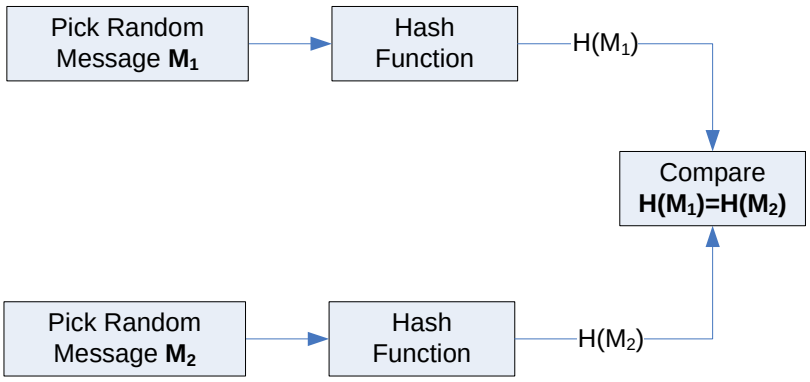


Figure 9 : Collision resistant property of Hash Function

Since hash functions are collision resistant, hash of a document can be used as a cryptographic equivalent of the document. This notion is used signing a digital document, for non-repudiation, rather than encrypting whole document with the private key of the sender which can be extremely slow, it is efficient and also sufficient to encrypt hash value of document with the private key of sender instead. Although primary usage of a one-way hash function is generating digital signatures, it can have other practical applications as well, such as storing passwords in a user database without divulging password or creating a file identification system and in our thesis creating secure audit log.

II.2.7 Message Authentication Code

To make the message secret and protect message from unauthorized disclosure, encryption is a good solution but how can be known that the message is not modified (i.e. integrity) or origin of the message (authentication) [29]. Let's explain the problem in an example: Assume that A and B, two entities, are sharing

a secret key and trusting each other on not divulging key. If entity A sends an encrypted message to entity B, by decrypting into a valid plaintext message, entity B can be in no doubt that the encrypted message did indeed come from the entity A, since entity A is the one who only knows the secret key and without knowing secret key, it is not possible to create a valid ciphertext. The only question that remains here for entity B is what "valid plaintext message" is, in other words how can entity B prove the validity of the message?

One solution for this problem is using hash functions. Entity A calculates a cryptographic hash value from the plaintext message and sends this hash value together with message (encrypted or non-encrypted, since we are discussing integrity and authentication of the message) to B. As long as the message remains unmodified (i.e., valid), its hash function can be recalculated and compared against the original hash value. If the hash value has not changed, the message has most likely not been altered is exceedingly high. Of course, it is not really sufficient to just calculate this hash value, since an adversary can also calculate hash functions at will, allowing that person to modify the message along with a brand new hash value. This would not effectively support either integrity or authentication. However, if A first calculates the hash value and then encrypts that hash value with a secret key, then this becomes a very effective solution. Message Authentication Code or MAC is obtained by applying a secret key to the message digest so that only the holder of the secret key can compute the MAC from the digest and hence, the message. Authentication Codes (MAC), and like hash functions produce a fixed sized output called a message *tag [15]*. This method thwarts the threat posed by a malicious interceptor who could modify the message and replace the digest with the digest of the modified message, for the interceptor won't have access to the secret key. Surely, there has to be a secure way to share the secret key between the sender and the recipient for this to work.

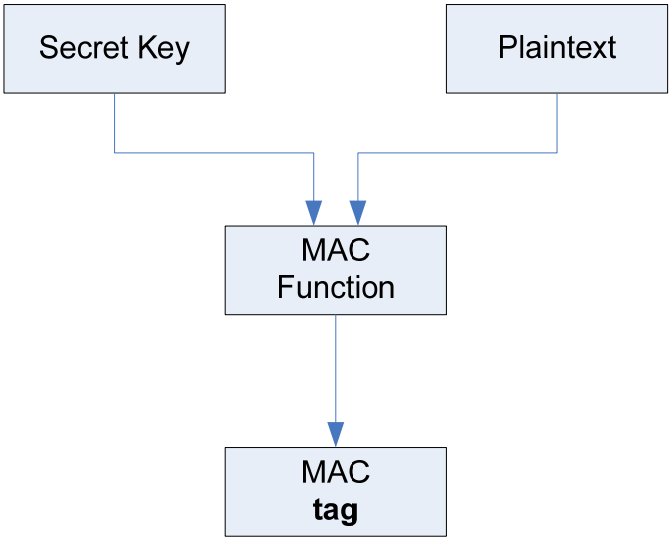


Figure 10 : MAC Function

CHAPTER III

RELATED WORK

Related works can be discussed in two parts: In the first part, studies focused on the integrity and confidentiality protection of logs will be examined. In the second part, retrieval of information from the encrypted logs and data will be discussed.

III.1 Secure Log

In 1997, Bellare and Yee have put forward a novel security property named as "Forward Integrity". Their main motivation was to find out a solution for the "mail sorter fraud problem" that is about outsourcing of mail collection from other organizations (e.g. hospitals, universities, local governments, etc). Also, they stated that the mechanism can be used by the logging systems to confirm or rebuke allegations of log records' modification before the moment of system compromise [30].

Then in [31, 32, 33], Schneier & Kelsey have proposed a secure logging scheme based on forward integrity concept for generating and verifying secure audit logs. Their system involves an untrusted host and a trusted machine which is capable of detecting any alterations and deletions on the logs which are produced before the compromise. To accomplish this, the system first establishes an authentication key

A_0 with a trusted third party before starting the logging operations. This initial authentication key evolves via one way function and overwrites by irretrievably deleting the previous value ($A_1 = Hash(A_0)$, $A_2 = Hash(A_1)$,…,$A_{j+1} = Hash(A_j)$). These evolving keys are used to calculate encryption key (K) in conjunction with each log entry type (W) such as system logs. ($K_1=Hash(A_1;W_1)$, $K_2=Hash(A_2;W_2)$,…, $K_{j+1} = Hash(A_j;W_j)$). Encryption keys are used for encrypting data (D) of each log entry ($E_{\mathbf{K}j}(D_j)$).

In order to let a semi trusted third party be able to verify the logs, a linear hash chain (Y) is constructed by hashing previous hash value of each log entry and concatenating some values of current log entry. ($Y_j = Hash(Y_{j-1};E_{\mathbf{K}j}(D_j);W_j)$).

Message Authentication Code (MAC) which protects the log entries against any modification is used. To compute Forward Integrity MAC, encryption key is not used but evolving authentication key ($Z_j = MAC_{\mathbf{A}j}(E_{\mathbf{K}j}(D_j))$). By this way, only trusted third party, who holds initial authentication key, can verify the audit log and since each record encrypted with different key, the trusted third party can decrypt particular logs selectively.

A log entry in Schneier and Kelsey scheme is like;

$\mathbf{W_j}$	$\mathbf{(E_{Kj}(D_j)}$	$\mathbf{Y_j}$	$\mathbf{Z_j}$

In [34], Chong, Peng and Hartel have tried to implement Schneier and Kelsey's secure audit logging protocol on tamper resistant hardware, iButton. Their implementation works both offline and online and also utilizes unforgeable time stamps in order to enhance the security but main problem of their scheme is performance decrease when compared to Schneier and Kelsey scheme. Performance evaluation of their system have revealed that it is not practical for logging systems which produce logs frequently such as internet access logs.

Therefore, they advised that usage of this system would be feasible for a system that only needs to log the main events such as playing a 4-minute song.

R. Accorsi and A. Hohl have worked on implementing secure logging scheme in pervasive computing systems. Their main goal is to distribute the logging task between resource rich devices which are trusted collectors and resource poor devices in the untrusted pervasive computing systems still protecting the logs. Therefore, they have proposed a secure logging protocol which is an adaptation of Schneier and Kelsey's secure audit logging protocol to pervasive systems [35].

III.2 Search on Encrypted Logs and Data

Encryption of stored logs is the straightforward solution for the confidentiality, but not without challenges foremost of which is the issue of searchability of encrypted logs. Searchable encrypted logs enable investigators search for a specific condition.

Waters *et al.* proposed two different schemes to provide searchability of the encrypted audit logs [36]. One of them is based on symmetric encryption and the other is asymmetric encryption which uses identity based encryption. In both schemes, a set of keywords are extracted before the log record is encrypted by a random symmetric key and these extracted keywords are used for search operation. In asymmetric encryption scheme, each keyword is constructed as a public key using identity based encryption. This IBE public key is used to encrypt symmetric key together with a flag. When an investigator wants to search for a keyword *w*, he needs to obtain a search capability from escrow agent for keyword *w*. If escrow agent authenticates investigator, it constructs search capability for keyword *w* and sends this capability to the investigator. Capability corresponds to private key in the IBE. Investigator controls the flag for each encrypted symmetric key, flag pair by trying to decrypt using capability. If there is a match, then random symmetric key is extracted and the log record is decrypted.

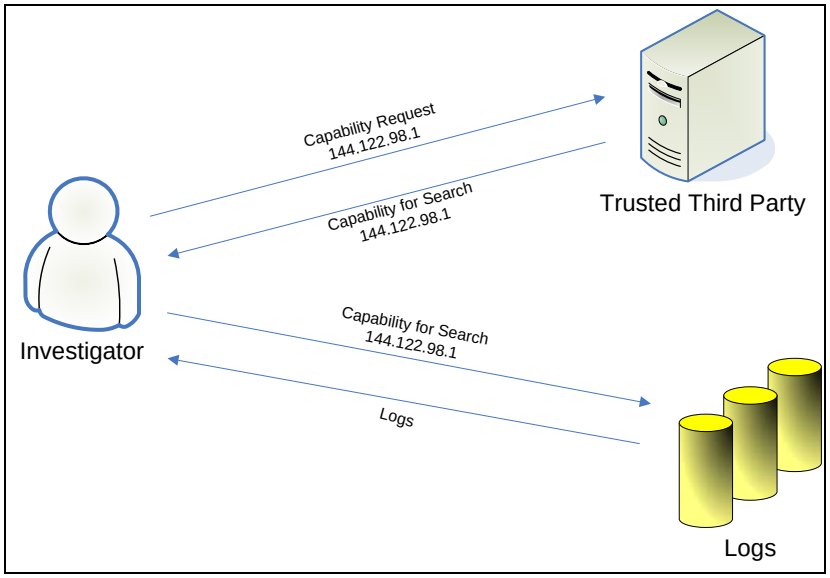


Figure 11: Search operation of an investigator

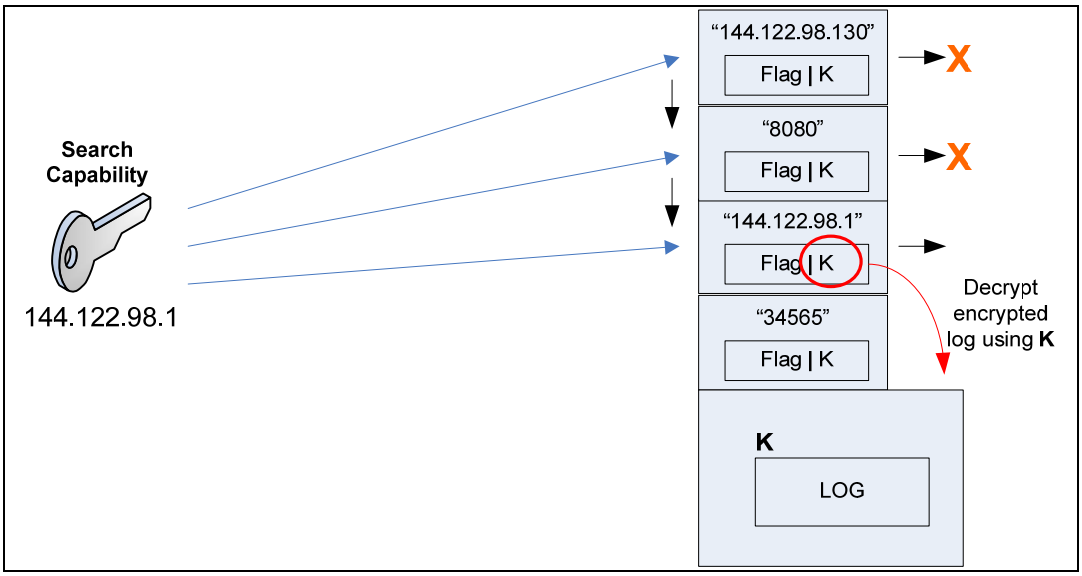


Figure 12 : Usage of capability

Waters *et al.* proposed an index based solution as an optimization of their main scheme to increase the speed of search operation. In this index based solution, logs are collected into "blocks" such as a block can include "t" line of logs. Keywords

of the logs in the blocks are extracted to build an index for each block. An encrypted log block and indexes are created as follows:

1. For the "t" line of logs, random symmetric encryption keys, $K_1 \ldots K_t$, are created for one time use.
2. Each log entry is encrypted using K_i
3. To build an index for each block, indices are constructed for each distinct keyword.
4. Identity Based Encryption is calculated for each keyword indices together with random symmetric keys.
5. Encrypted logs and index are written to the log file.

We will cover this topic in detail in Chapter 4.

A time-scoped search is another concept which has been proposed for encrypted audit logs by Davis et al. in 2004. It is basically based on the idea of Waters et al. For each keyword, they construct a back pointer which shows in which logged record that keyword is lastly included and storing back pointer for each keyword enables an investigator to make time scoped searching. At the same time, anchor logs are created to define time interval in other words boundaries of records are delineated. Anchor logs limit the search operation to precisely the time periods. Therefore, an investigator gains no information for other time periods and other keywords. They success restricted delegation of searches on encrypted audit logs, since boundaries of log records are described [37].

In 2005, Ohtaki described an index base solution to search for encrypted logs by the idea of plain text search which uses inverted index. Inverted index stores a list of record identifier (possibly the location) of a set of words in the records. B-tree or heap can be used storing indexes Ohtaki built encrypted inverted index which enables investigator to search for a given keyword. In their encrypted inverted index scheme an encrypted linear list is used to hold set of location where the keyword appear. Each list item which is encrypted includes log entry identifier I_i

and a pointer to next list item. For each word, a label is calculated and an index entry with this label points the head of each linear list. Since it uses an index based approach, response time of their scheme for a search is very high when compared to non index based solutions [38].

Some other works for making search on encrypted data have also been carried out previously. But they have not specifically focused on encrypted audit logs. The majority of previous works on this topic have focused on the users who store their data (e.g. e-mail) on an untrusted server and enable them to selectively gather their data without revealing the content. The question on how to perform searching on encrypted data was raised originally in [39] and a scheme for searching for sequences of words based on stream cipher in a symmetric-key setting was proposed. Goh used Bloom filters to make an efficient scheme for keyword search over encrypted data [40]. Also [41, 42] defined index based solutions. Public key schemes for keyword search over encrypted data are presented in [43]. As it can be understood from all these previous studies, work on searching encrypted data has largely focused on search criteria consisting of a single keyword. Later on, secure conjunctive keyword search over encrypted data was proposed in [44, 45].

III.3 Discussion on Related Work

To provide the authenticity of the log records, it is needed to make them verifiable. The scheme that Schneier and Kelsey have proposed was a good one for providing authenticity and verifiability. Then various subsequent studies have followed the same idea. Broadly speaking, verification should be considered in two parts [36]:

1- In the case where deletion or any kind of alteration is of concern, individual log verification should be provided to make the remaining part of the logs useful.

2- Logs should also be linked to each other in order to determine the missing part strictly. In other words whole log verification should be provided. The main problem of Schneier and Kelsey's scheme is that it depends on the previous values to detect any anomaly but when one log is deleted from the chain, it is not possible verify to remaining log whether is altered or not. This brings about the problem of authenticity of the logs following the deleted record. In other words, their scheme does not provide individual log verification.

As the logs are the main source for digital security, providing integrity, privacy and authenticity of the logs is an issue of paramount importance. We know that log records are used to detect and comprehend damages of a computer or network system caused by not only intrusions but also defects or accidents. Therefore, logs which come from different sources in different formats should be scanned carefully in order to determine whether they include some specific patterns or not. [30, 31, 32, 33, 34, 35] have overlooked this issue and they have only focused on a system which is capable of detecting any alterations and deletions on logs.

Most of the evidences are ferret out from logged actions that can be used to prove whether an agent has performed a specific activity or not. Therefore, in the time of trial, authenticity of the logs should definitely be provided. While studies in the first part focused on the integrity protection on the logs, subsequent studies on the logs focused on searchability of encrypted audit logs not too much considering on integrity protection of logs. Also, their schemes work for only single keyword search.

Besides studies focused specifically searching on encrypted logs, there are other studies which investigate searching on any encrypted data. These studies mostly try to solve problem of encrypted document (e.g. e-mails) retrieval from an untrusted server. In an example, their schemes work as follows: If the user wants to retrieve his documents, first a certain search criterion is defined. For the e-mail example, "subject", "name", "to" parts are used for search criterion and they are

known by the user but not by the server because both documents and keywords are encrypted before the documents are put in the server. To determine which documents contain a specific keyword, server requires a piece of information called a capability for keyword. Using capability, server retrieves documents containing only the given keyword and gathers no other information. In other words, without capability for a keyword, server is not able to get any information about documents. Again, solutions for the problem of searching on encrypted data have mostly focused on single keyword search. It is not a perfect solution since documents are belonging to a user, he knows for what he is searching. Therefore, he can reach required document. For example, in the e-mail case, if Alice needs to learn whether there are e-mails from Bob or not, she knows that Bob is a search criterion. If Alice and Bob are close friends and they send email each other too many, there may be too many match for this criterion but it is not comparable to a single keyword search for encrypted logs because there may be thousands of match for a single keyword. Single keyword search over encrypted logs may result too much information and most of which are not necessary.

For the same problem setting, conjunctive keyword search on encrypted data schemes are proposed [44, 45]. But they assume that user knows exactly what he is looking for. In the previous scheme, Alice can search for e-mail from Bob, but in conjunctive keyword search scheme, she should search for From:Bob, Subject: my love Date: 02.14.2007. Clearly their assumptions are:

- The same keyword never appears in two different fields for the same document. For example, To, From parts of an e-mail can not be Bob at the same time.

- Every keyword field for a document should be defined. If user did not define it, a special keyword is assigned. For example, if the Subject field is not defined by the user, “NULL” word can be assigned.

Investigators can search log using different keyword fields of the log. Sometimes investigator use only source IP address, sometimes source IP and destination IP or he can make some other combinations using other keyword fields. As conjunctive keyword schemes strictly impose that every keyword field must be used on search operation, they are not so practical for logs. As stated above, investigator usually does not know all the fields he is searching for. Fields of the log are actually the data of the log itself, therefore investigator makes some combinations on the keyword field to learn whether other keyword fields include the data he is looking for or not.

Although response time of Ohtaki's index base solution for a search is very high when compared to non index based solutions, problem they try to solve is different from ours. In their problem, administrator of a computer system who is trusted discloses logs in encrypted format to an investigator who is police in their case. Administrator stores log in a plain text format but if police asks to search logs, to protect the privacy of innocent users, logs are converted in an searchable encrypted format.

Another critical issue in the logs is that; sometimes statically analyzing logs is needed rather than making search and finding a specific record. Searching based on a keyword may not give always enough information to understand the system. Most of the time, to understand what the problem is, it is needed to correlate logs from different resources and make decisions on this correlation. The schemes we have mentioned previously do not address these issues.

CHAPTER IV

PROBLEM DEFINITION AND OUR PROPOSED SOLUTION

IV.1 Problem Definition

Logging is one of the central services in computing systems. It gathers and stores events occurred in the systems. Since logs include incriminating evidence such as proofs of intrusions into the system, after a security breach, logs are the main targets for attacks. Although they contain such valuable information, mostly, logs are kept as sequential entries to a plain text file and protection of this file provided by the underlying operating system. Once the operating system has been compromised, the logs are at the hand of the attacker. In such a model, logs are authentic, if the system has not been compromised.

In our problem setting, there is an untrusted machine which can not be guaranteed that it can not be compromised. This untrusted machine can be a computer or any kind of network device that maintains a file of log entries of users' network activities. Even in the event that an attacker takes over this logging machine, we want to assure that authenticity of logs is preserved. More precisely, if a logging machine is captured at time *t* by an attacker, we want to provide security to logs

until time *t*. An attacker who gains control of logging machine at time *t* can not read or modify log entries before time *t*. If a modification occurs on log entries which are produced before time *t*, we want to detect this modification.

Briefly, we want that our mechanism provide;

1. The attacker cannot modify log file undetectably; can not insert fake log entries or delete log entries.
2. The attacker cannot see content of log entries, and thus cannot ensure whether the log file include information that will be used as evidence in the time of trial.

To be able to identify incidents such as a host being infected by malware or a person gaining unauthorized access to a host, logs must be searchable but encryption of log entries makes hard to retrieve data selectively. Inherited from the studies searching on encrypted data, studies searching on secure audit log focused on the single keyword search which often yields far too coarse results. For example, some logs include source and destination IP addresses. If an investigator wants to search for logs which include xxx.xxx.xxx.xxx source IP address, he will get logs including that IP not only in the source fields but also in the destination fields. This increases the rate of unnecessary results in the search operation. As we discuss previously, there is not adequate work on analyzing secure audit log. Some conjunctive keyword search schemes have been proposed on encrypted data but investigator has to write all fields when searching for a document. Therefore, conjunctive keyword search schemes are not practical for analyzing encrypted logs. In addition, since their main focus of previous studies is searching on encrypted logs, they could not give enough attention to security of logs but security and searchability are the twin requirements. Without one of these requirements, the other will not be useful; if an investigator can not search and analyze the logs, it is not important whether the logs are secure or not because investigator can not gather required information. If the authenticity of logs can not be proved in the time of trial, it is not important how fast and secure information is

gathered from the encrypted logs. As a result we can say that, if these two requirements are not provided, usefulness of the logs is lost.

Fundamentally, problem arises from the problem setting of searching on encrypted data. Because most of the schemes deal with encrypted document retrieval from untrusted server rather than focusing on encrypted logs. Therefore, the aim of these studies is to find the related document which is most of the time known by the user and can be created offline without a time restriction, but in our problem setting, we also deal with the information which has not been known before until gathered and examined.

As can be understood from above discussion, simultaneously protecting the integrity of the log, controlling access to contents, and maintaining its usefulness by making it searchable are the main challenges to build a successful secure logging mechanism. By considering these we, in this thesis, build a secure and searchable audit logging mechanism.

IV.2 Design Requirements for Secure Audit Log

We can identify five important design requirements for logs increasingly used as "audit logs" [46]:

I- Integrity:

Integrity means completeness of a message or messages. Log is composed of log entries; each entry includes information related to a specific event that has occurred within a system or network. Therefore, once an attacker has gained access into the system, he tries to modify logs to cover up his tracks and to avoid detection by system, network, and security administrators. Attacker can completely purge log files or alter logs a line-by-line basis to keep normal system events while

cleaning suspicious log entries. In the first case, completely removing log entries from log files is an indication of a probable intrusion. Hence, attackers prefer to alter particular events from the logs associated with the attacker's gaining access, elevating privileges, and installing back doors. To establish effective security, providing integrity for logs that represent the entire history of the incident is critical [9].

II- Authenticity:

"Electronic documents will only stand up in court if the who, what, and when they represent are unassailable" [47]. As the log files are also an electronic document and it has to be proved that it can be used as evidence in the court. In order to guarantee the legal validity of the logs, authenticity is one of the prerequisites.

III- Confidentiality:

Logs contain diverse information and usage of this information by unauthorized persons can harm one's privacy. This raises security and privacy concerns and confidentiality provides that the information in logs is not disclosed to unauthorized persons or processes [46].

IV- Verifiability:

If authenticity and integrity of logs are provided, it must also be verifiable so that the time, date, and content of that log are not changed after it was created. Verification can either be done publicly or via trusted verifier. In the former case, anyone who has proper public information is able to verify logs. In the latter case, there is a need for trusted party who keep one or more secrets [36].

V- Searchability:

As stated above, logs contain a wide variety of information. Relevant logs should be efficiently searchable by authorized persons but it should be impossible for that person to reach other special information that is out of concern. For example, if an investigator is delegated to search for access patterns of a specific user, he could not be able to get financial data of that person from logs [36].

IV.3 Our Proposed Solution

Our method comprises untrusted host such as firewall, router or any untrusted host, semi-trusted log storage server and a trusted third party.

IV.3.1 Log Entry Definitions

1- $R_j \rightarrow w_j^{(1)}, w_j^{(2)} \ldots w_j^{(m)}$:

Log file consists of Records (R) and each record includes keywords ($w_j^{(1)}, w_j^{(2)}, \ldots, w_j^{(m)}$). *j* defines the index number of a log record and *1,2, …, m* are the numbers of keyword in the R. Often, the log format is predetermined. Therefore, as a matter of fact, these keywords construct the record. In document retrieval case, keywords are manually determined by the user or extracted using some characteristics of document and these keywords define the document but in the log case, keywords themselves are data itself and there are standards for syslog, HTTP logs, and many others.

2- A_j → Authentication key:

A_j is the authentication key of MAC for jth entry in the log. Untrusted machine should have A_0 before start producing the log file.

3- RSK_j → Random Symmetric Key:

This Random Symmetric Key will be used for encrypting a log entry (R_j). RSK_j will be used one time.

4- E_{RSK} → Encrypt with Random Symmetric Key:

Due to privacy reasons R_j must be encrypted in a way that only authorized persons can access the content of a log record.

5- Y_j → Hash chain:

A chain of hash values of previous log entries (R) ties the log stream together. Although this hash chain is not acceptable as an authentic evidence in the time of trial, it enables semi-trusted third party verify the log file. Y_j is calculated using encrypted log records so that semi trusted third party can verify it without knowing the content of the log.

6- Z_j → MAC_{Aj} (Y_j):

Calculating the MAC of hash chain parts helps us to link each log record to the next one. By this way we can determine the missing part strictly and prove any alteration. In other words, we provide whole log verification.

7- P_j → MAC_{Aj} (Single log)

In the case where deletion or any kind of alteration is of concern, individual log verification should be provided to make the remaining part of the logs useful.

8- A_{j+1} → Hash ("Increment Hash", A_j)

Authentication key is hashed after a log is created and the previous value of it is irretrievably deleted. In this manner, if untrusted host is compromised, the adversary is not able to get the authentication key of the previous log entries. Therefore, he could not success to modify the previous log entries without being detected.

9- T_j → Log type:

There are different types of logs and each of them serves for different purposes. Therefore, if someone is given permission to analyze the logs, it is important to take into consideration which type of logs will be accessible for that specific user.

10- ibePK → (T_j |w_j) Identity Based Encryption Public Key:

This key is going to be used for encrypting the random symmetric key together with w_j. This ibePK will be calculated for each keyword concatenating with log type T_j.

11- $C_j^{(1)}$ → $E_{ibePKj}^{(1)}(w_j^{(1)}|RSK_j)$:

C denotes ciphertext of the Identity Based Encryption.

12- ibeSK → Identity Based Encryption Secret Key:

ibeSK is the corresponding private key for ibePK.

IV.3.2 How Scheme Works

The figure below shows the construction of jth secure analyzable audit log entry from a jth log record.

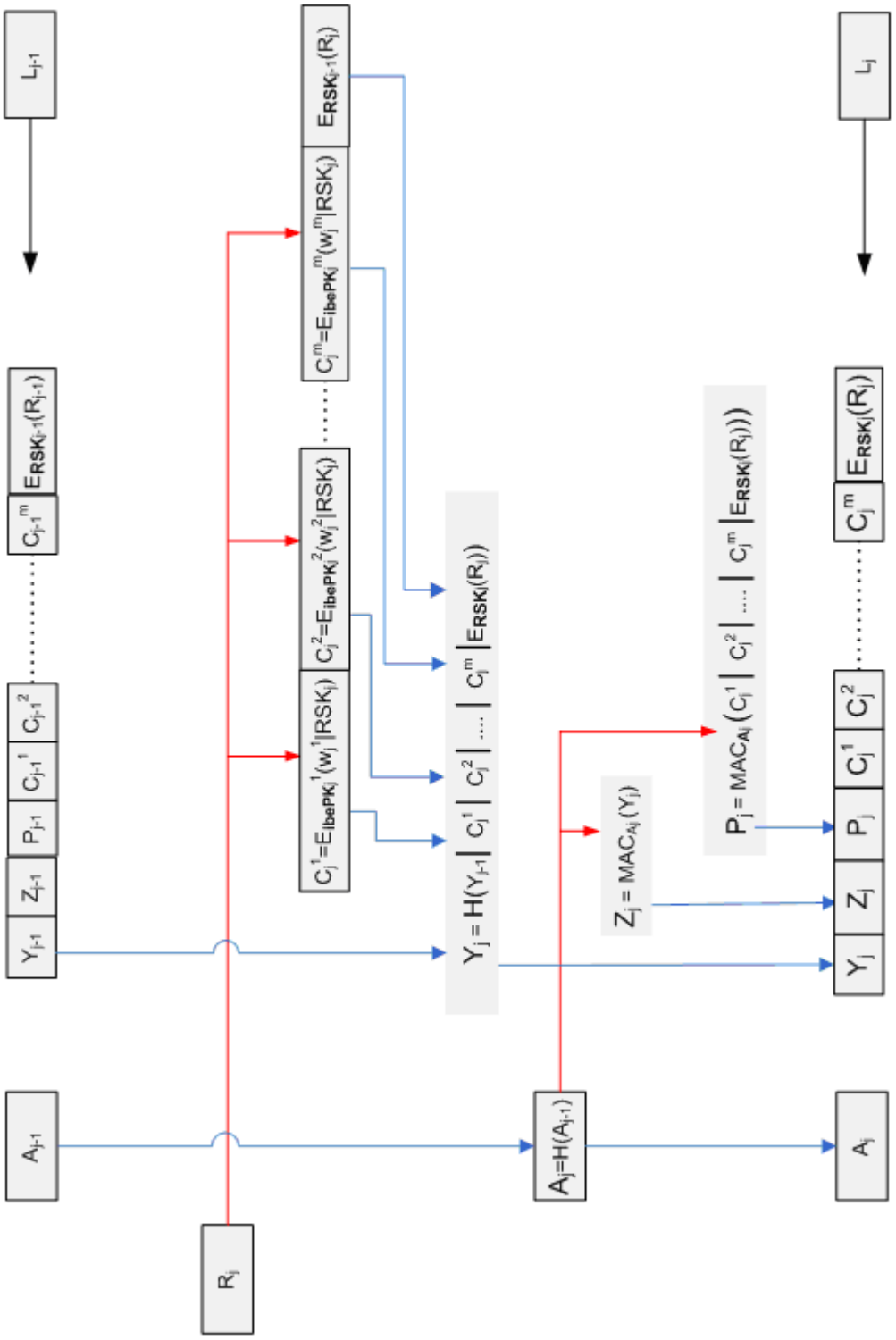


Figure 13 : How Scheme Works

IV.3.2.1 Initializing Log File

Before the log records is written to log file, A_0 must be established between the trusted third party and untrusted host. After this authentication key, A_0, is established, an initial start-of-log message is written. This initial seed value will be used for forward security. Each time a log produced, this value is going to be used to calculate MAC of current encrypted values together with the previous hash.

$$Y_j = H\left(Y_{j-1} | E_{ibePK\ j}^{(1)}(w_j^{(1)} | RSK_j) | E_{ibePK\ j}^{(2)}(w_j^{(2)} | RSK_j) | \ldots | E_{ibePK\ j}^{(m)}(w_j^{(m)} | RSK_j) | E_{RSK_j}(R)\right),$$

$$Z_j = MAC_{Aj}(Y_j)$$

By using this MAC, we are able to determine any deletion, or alteration without any doubt. Also this seed value is going to be used for the single log verification which is not provided by the Schneier and Kelsey mechanism.

To be able to verify each log record individually, we calculate the MAC of hash which is the message digest of current encrypted values with the key A_j.

$$P_j = MAC_{Aj}\left(E_{ibePK\ j}^{(1)}(w_j^{(1)} | RSK_j) | E_{ibePK\ j}^{(2)}(w_j^{(2)} | RSK_j) | \ldots | E_{ibePK_j}^{(m)}(w_j^{(m)} | RSK_j) | E_{RSK_j}(R)\right)$$

Following these operations, MAC key is derived by using one way process (A_j=H (A_{j-1})) for the next log. The new value of the MAC key overwrites and irretrievably deletes the previous value.

Note that, since we have used Identity Based Encryption public key to encrypt each keyword, we do not need to establish any public key before the log operation start but we only need the public parameters of trusted third party. In Schneier and Kelsey's mechanism, symmetric encryption is used. The problem of symmetric key is that same key is used both for encryption and decryption. To solve this

problem, they use different symmetric keys to encrypt each log entry and by this way, they prevent decryption of previous encrypted entries generated before the intrusion. In public key case, even an attacker have taken the control of the untrusted machine, he will not be able to decrypt the previous entries, because he has to know the private key which are generated by the trusted third party.

IV.3.2.2 Construction of jth Log Record

Suppose that jth log record (R_j) consists of keywords, $w_j^{(1)}, w_j^{(2)}, \ldots, w_j^{(m)}$.

At the time a log entry is written, the following steps are performed:

i) Authentication key of previous log record (A_{j-1}) is hashed to get authentication key of *jth* log record and A_{j-1} is irrecoverably deleted.

$A_j = H(A_{j-1})$

ii) Untrusted host chooses random symmetric key (RSK_j) to be used only for this entry. Then untrusted host encrypts R_j using RSK.

$E_{\mathbf{RSK}_j}(R_j)$

iii) For each keyword, $w_j^{(1)}, w_j^{(2)}, \ldots, w_j^{(m)}$, untrusted host computes the Identity Based Encryption Public Key using public parameters of trusted third party.

$ibePK_j^{(1)} = (T_j \mid w_j^{(1)})$

$ibePK_j^{(2)} = (T_j \mid w_j^{(2)})$

.

.

.

$ibePK_j^{(m)} = (T_j \mid w_j^{(m)})$

iii) Using Identity Based public keys for each keyword, untrusted host encrypts RSK_j concatenating with keyword one by one.

$C_j^{(1)} = E_{ibePK_j^{(1)}}(w_j^{(1)} \mid RSK_j)$

$C_j^{(2)} = E_{ibePK_j^{(2)}}(w_j^{(2)} \mid RSK_j)$

.

.

.

$C_j^{(m)} = E_{ibePK(j^m)}(w_j^{(m)} RSK_j)$

It is worth noting that encrypting keywords with this scheme does not leak any statistical information. If we simply encrypt each keyword of R_j without concatenating with RSK_j using IBE public key then the results can be as followings:

bS9jcy9jcnlwdG8	juMlnUx2Am8St	uaXRlRmllbGRQ	G42F4CAAJMAAF4
L0JpZ0ludGVnZXI	lbGR0AChMbnVp	pbml0ZUZpZWxk	aWVsZHyn52s9lD
cmltZQpq7v3QT15	DeHBzcgAibnVp	0ei5maWVsZC5G	ABBqYXZhLmxh4i
w5Sbk2q87UoCAAa	lbGR0AChMbnVp	ubWF0aC5CaWdJ	aErMW13zZ90dFs
bS9jcy9jcnlwdG8	MamF2YS9tYXRo	0Tm9uemVyb0J5	hyACBudWltLmNz

Figure 14 : Using same public key

Since some values in logs are the same, after encrypting these values with IBE public key, the results will be the same. Also, values of parts in the logs have mostly limited range. For example, port numbers' values are between 0 and 65535. Someone encrypting all values of ports with public key and comparing these values with the encrypted ones in the logs can understand what the current port is. To prevent this, values of keywords in the log records are encrypted with RSK_j.

Since RSK_j is randomly chosen for each log record, we prevent such an information leakage.

iv) To enable semi-trusted third party verify logs, message digest of encrypted values of R_j concatenating with (Y_{j-1}) is calculated.

$$Y_j = H(Y_{j-1} | C_j^{(1)} | C_j^{(2)} | .... | C_j^{(m)} | E_{RSKj}(R_j))$$

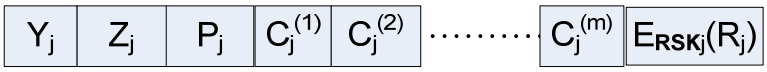
v) To provide forward security, MAC is conducted using hash of encrypted values of R_j and previous one using A_j as a key.

$$Z_j = MAC_{Aj}(Y_j)$$

vi) To provide single log verification, MAC is conducted using hash of encrypted values of R_j using A_j as a key.

$$P_j = MAC_{Aj}(H(Y_{j-1} | C_j^{(1)} | C_j^{(2)} | .... | C_j^{(m)} | E_{RSKj}(R_j)))$$

vii) jth analyzable log entry is written to the log file.



IV.3.3 Validation of Log Records by Semi-Trusted Log Storage Server

When semi-trusted log storage server receives the complete secure log, he can validate it using hash chain field without knowing content of the log records and initial authentication key. The hash chain field contains a hash of the encrypted payload of current log record, as well as the previous log entry's hash chain field. Semi-trusted log storage server with access to the encrypted log entries can verify

the hashes to detect modifications to the logs. Removing or inserting a single log entry in this chain invalidates the verification hash value because there would not be a match between what is expected in the field and what is written.

IV.3.4 Analyzing and Decrypting Encrypted Logs

There are different types of logs such as accounting, network traffic, logins and logouts, and dozens more. If these logs can not be analyzed effectively, building a secure audit logging scheme will not be useful. Especially to identify an attacker, clues which reveal the identity of the attacker must be captured. Therefore, investigation in depth must be conducted. Most of the logs have a specific format and we construct our encrypted log scheme by considering this. In our solution, fields of the log record can be investigated selectively. In other words, investigator can search log with a single keyword or multiple keywords.

Let's assume that an investigator wants to learn access list of an IP address. To be able to do this, he needs a capability for that IP address. As a matter of fact, this capability corresponds to private key of IBE encryption. Since, we encrypt RSK by concatenating with current keyword using IBE public key that is the concatenation of log type and keyword itself, capability will enable investigator decrypt the IP column of each log record and compare whether there is a match with that IP or not. Therefore, investigator, first, authenticates himself to the trusted third party in order to get capability for that IP address. If trusted third party approves investigator, it prepare a capability according to the access permission of the investigator to the logs. For example, if the investigator has only access permission to the internet access logs not other types of logs such as system logs, search capability will be constructed using type of internet access logs and IP address.

ibeSK will be calculated using ($T_{\mathbf{InternetAccessLogs}}$ | IP) and it corresponds to IBE public key ($T_{\mathbf{InternetAccessLogs}}$ | IP)

Suppose that we get a capability for an IP address in Internet access logs. Then search operation will be as following:

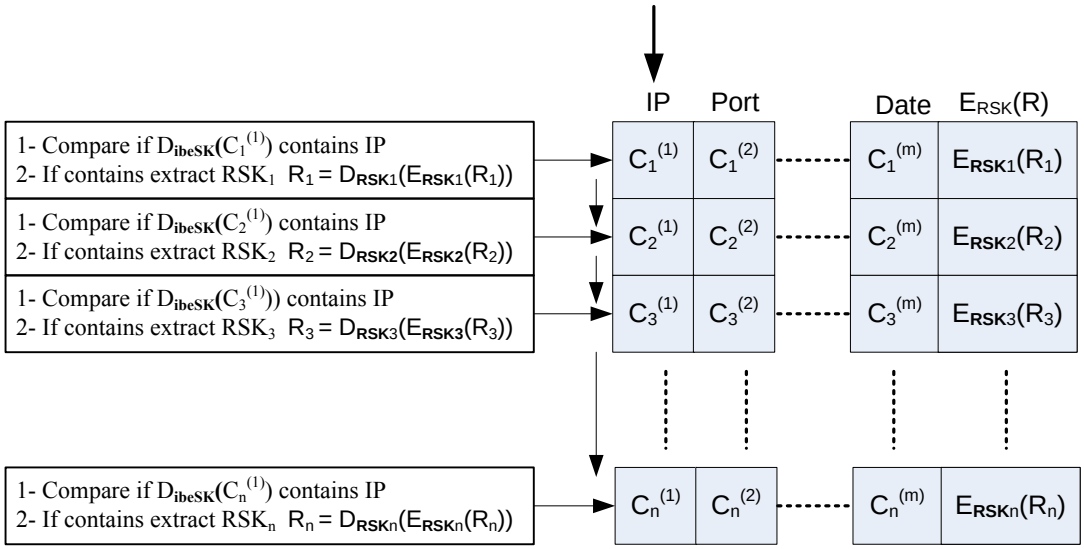


Figure 15 : Search operation using single keyword

At the same time, since we construct our mechanism by taking into consideration that logs have predetermined structure; the other parts can be searched by using the same mechanism. Let's assume that an investigator wants to learn access times of an IP address to a specific application. Mostly, each application in the internet uses a specific port. For example, web applications use port 80. To make this search, an investigator needs capability for the IP address and port number. Suppose that we get a capability for the IP address and port number. Then search operation will be as following:

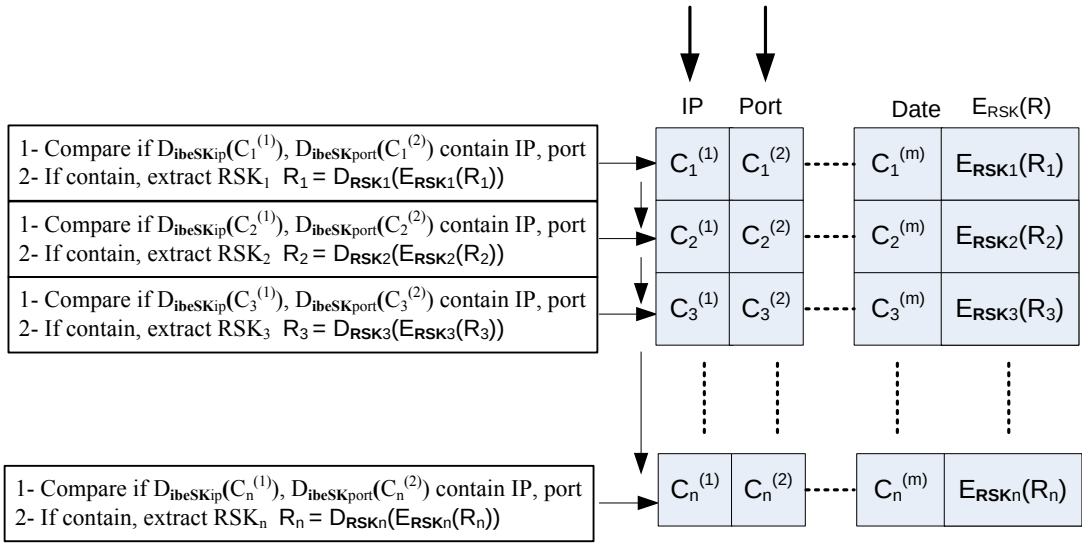


Figure 16 : Search operation using multiple keywords

IV.3.5 Comparison of Methods

To speed up search operation, Waters *et al.* proposed an index based solution. They did not implement their scheme, but basically describe how this scheme can speed up search operation. In this index based solution, before logs are encrypted and make searchable, original logs are collected into "blocks" such as a block can include "t" lines of log. Let's assume that a block contains "*t*" lines of log. An encrypted log block and indexes are created as follows:

1. Random symmetric encryption keys, $K_1 \ldots K_t$, are created for one time use.
2. Each log entry is encrypted using K_i

   $$E_{\mathbf{K_i}}(R_i)$$

3. To build an index for each block, indices are constructed for each distinct keyword. For example, a distinct keyword w_j in a block can have $\{i_{j,1}, \ldots, i_{j,N}\}$. N is the log entries which w_j belongs.

4. Identity Based Encryption is calculated for each keyword indices concatenating with random symmetric keys using each distinct keyword as a public key of identity based encryption.

   IBE Public Key= w_j

   $(flag \mid i_{j,1} \mid Ki_{j,1} \mid \ldots \mid i_{j,N} \mid Ki_{j,N})$ is encrypted using w_j.

   $cj = IBE_{w_j}(flag \mid i_{j,1} \mid Ki_{j,1} \mid \ldots \mid i_{j,N} \mid Ki_{j,N})$

5. Indexed log block that includes encrypted logs and index are written to the log file.

   $$c_1 \ldots c_u, E_{\mathbf{K_1}}(R_t), \ldots, E_{\mathbf{K_t}}(R_t)$$

Search operation is realized same as original scheme of Waters et al.: First investigator gathers a capability for interested keyword. Investigator uses this capability to decrypt each "***c***" values in a block. After decryption operation, decrypted value is compared with flag. If decrypted value matches with flag, indices and symmetric keys are extracted. Encrypted logs are decrypted using these symmetric keys.

Waters et al state that indexing provides a significant performance advantage for searching when keywords are repeated among several log entries within a block. We implement previous scheme which is non indexed and indexed scheme of Waters et al. For the implementation, we create 1000 lines of log and we make a search operation using two distinct keywords. One of these keywords, destination IP number **212.175.170.34,** is frequently repeated among log entries. The other, source IP number **62.121.66.223,** is less frequent when compared to the other one. To depict how block size effect total search time, we make search operation by increasing block size for the indexed version. We perform search operation more than once and below tables show test results of the total search times for indexed scheme of Waters et al.

Table 1 : Total search times for 62.121.66.223 in different block sizes and tests

	62.121.66.223									
	Block size 10	Block size 20	Block size 30	Block size 40	Block size 50	Block size 60	Block size 70	Block size 80	Block size 90	Block size 100
Test 1(sec)	5501186	3168174	2141620	1725812	1588032	1108338	754607	893702	847070	926624
Test 2(sec)	5511900	3191768	2123922	1725733	1579796	1089426	746439	883576	846615	935360
Test 3(sec)	5547261	3173348	2154135	1822076	1577298	1076910	737481	895796	854059	943595
Test 4(sec)	5554724	3142513	2132062	1756968	1581504	1084123	755742	875531	838726	922514
Test 5(sec)	5499263	3262108	2152248	1764765	1580577	1089585	752195	877086	846053	921548
Average(sec)	**5522867**	**3187582**	**2140797**	**1759071**	**1581441**	**1089677**	**749293**	**885138**	**846505**	**929928**

Table 2 : Total search times for 212.175.170.34 in different block sizes and tests

	212.175.170.34									
	Block size 10	Block size 20	Block size 30	Block size 40	Block size 50	Block size 60	Block size 70	Block size 80	Block size 90	Block size 100
Test 1(sec)	2599107	1342276	735939	678280	597016	428787	338072	398321	241945	374064
Test 2(sec)	2581370	1346409	751715	672621	610861	419681	339099	405752	246406	370455
Test 3(sec)	2607160	1308562	733110	676622	606624	417053	341440	383971	245598	375907
Test 4(sec)	2578124	1327408	732087	672404	595827	420099	342970	370671	247741	376466
Test 5(sec)	2604690	1336405	742786	671843	602860	417572	340148	373580	245322	377345
Average(sec)	**2594090**	**1332212**	**739127**	**674354**	**602638**	**420638**	**340346**	**386459**	**245403**	**374847**

Below figures show the total search times for Waters et al index based search.

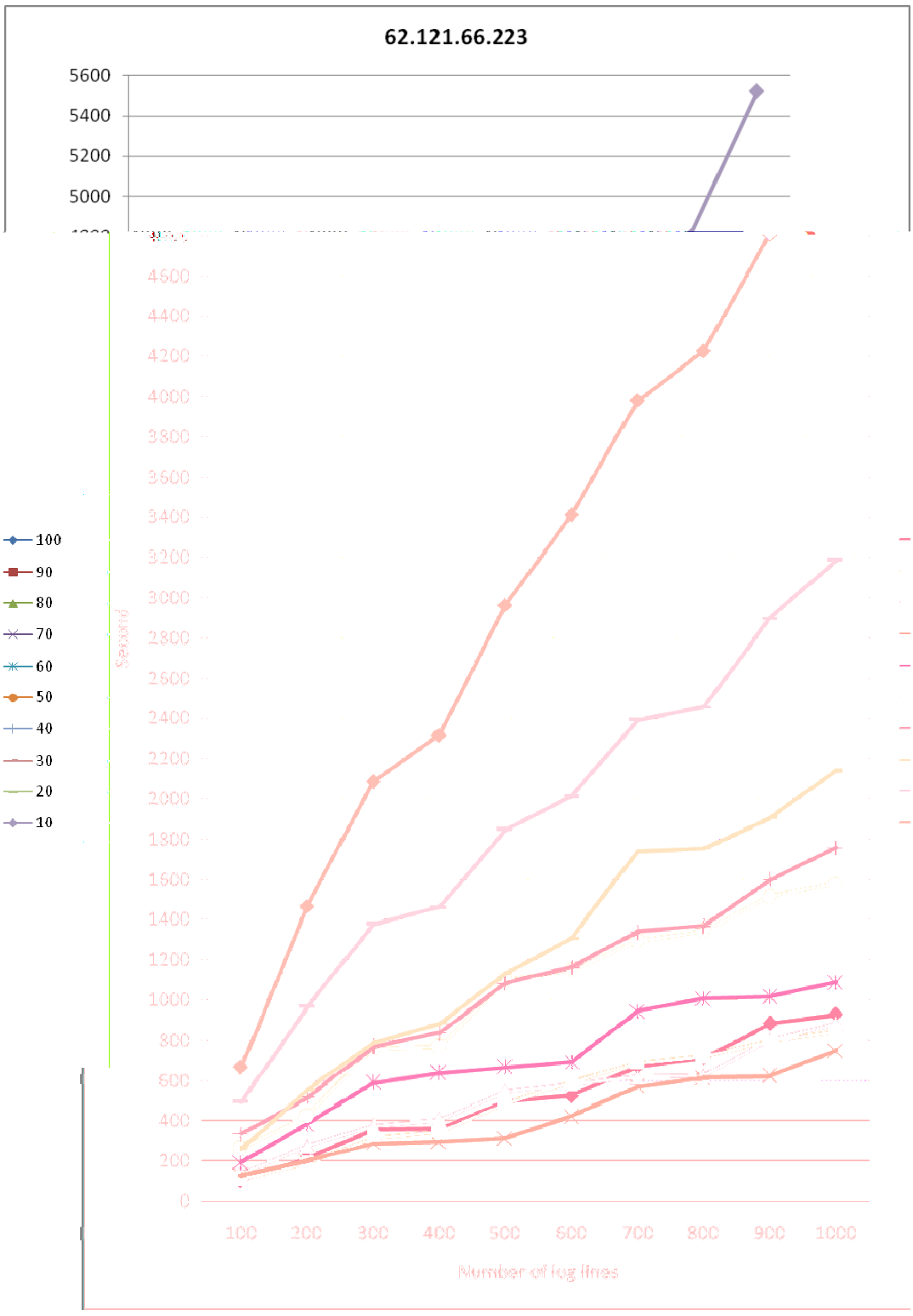


Figure 17: Total search times for Waters et al index based search using less frequent repeated keyword

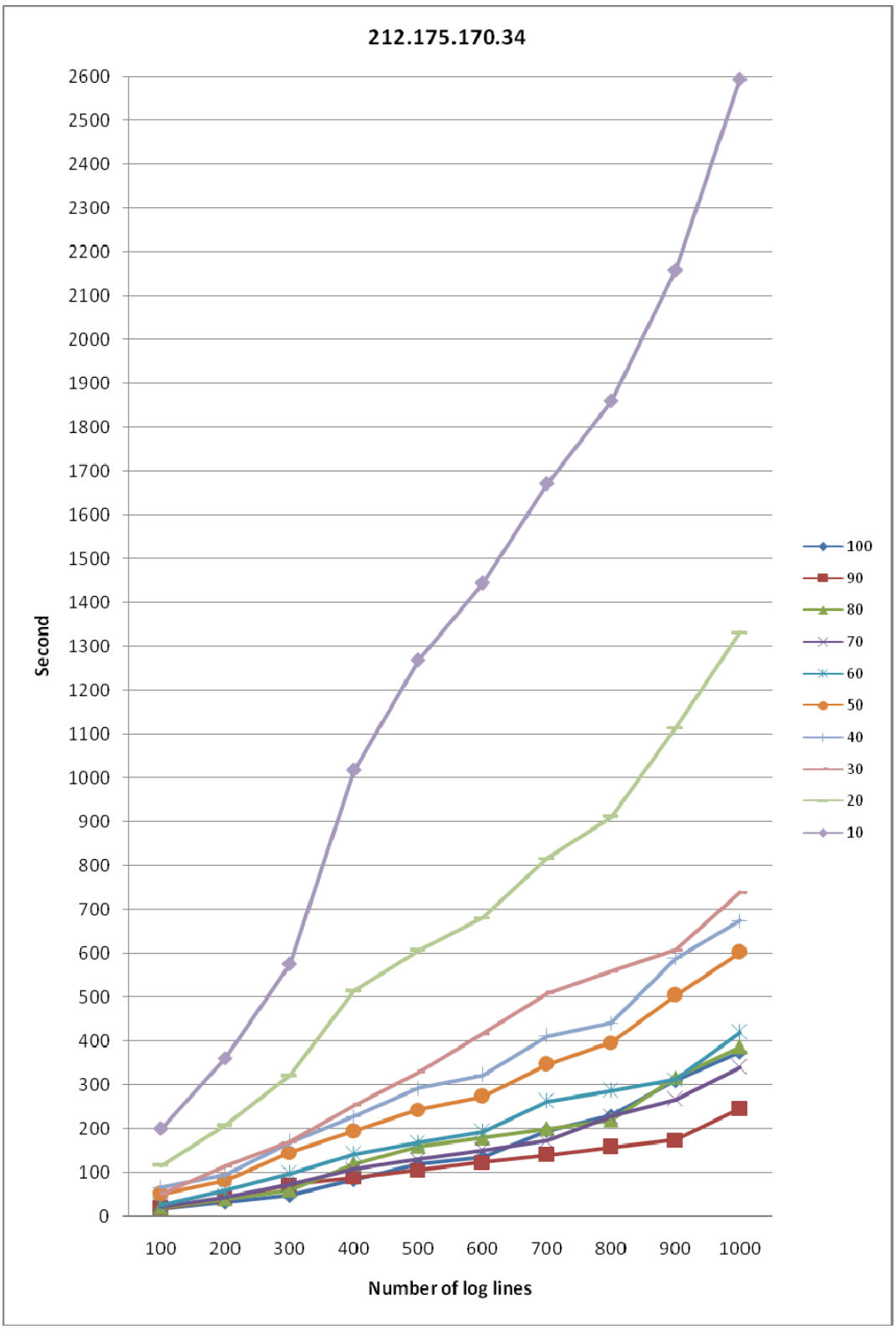


Figure 18 : Total search times for Waters et al index based search using frequently repeated keyword

Y axis of the above figures shows total search time as seconds and X axis shows the block size (number of log entries). As the figure depicts that total search time decreases when the searched keyword frequently repeated in the log entries. Also, at first while the block size decreases, sharp decreases are observed on the total search time. But after, increase in the block size does not much affect total search time. Although it is expected that increase in the block size results decrease on the total search time, the results of our implementation show that this expectation is not always true. The main reason for this consequence is that to find a match for searched keyword, it is needed to decrypt $c_1 \ldots c_u$ values linearly. Therefore, sometimes a keyword match can occur at the end of the $c_1 \ldots c_u$ values, this leads to increase in the total search time.

On the same logs, we implemented and tested non indexed scheme of Waters et al. We perform search operation more than once and below table show test results of the total search times for non indexed scheme of Waters et al.

Table 3 : Total search times for non indexed scheme of Waters et al in different tests

	62.121.66.223	**212.175.170.34**
Test 1	15981189	15834986
Test 2	15981319	15457468
Test 3	15906031	15441221
Test 4	15953640	15461322
Test 5	15912915	15849282
Average	**15947019**	**15608855.8**

Below figure shows total search times for the non indexed version.

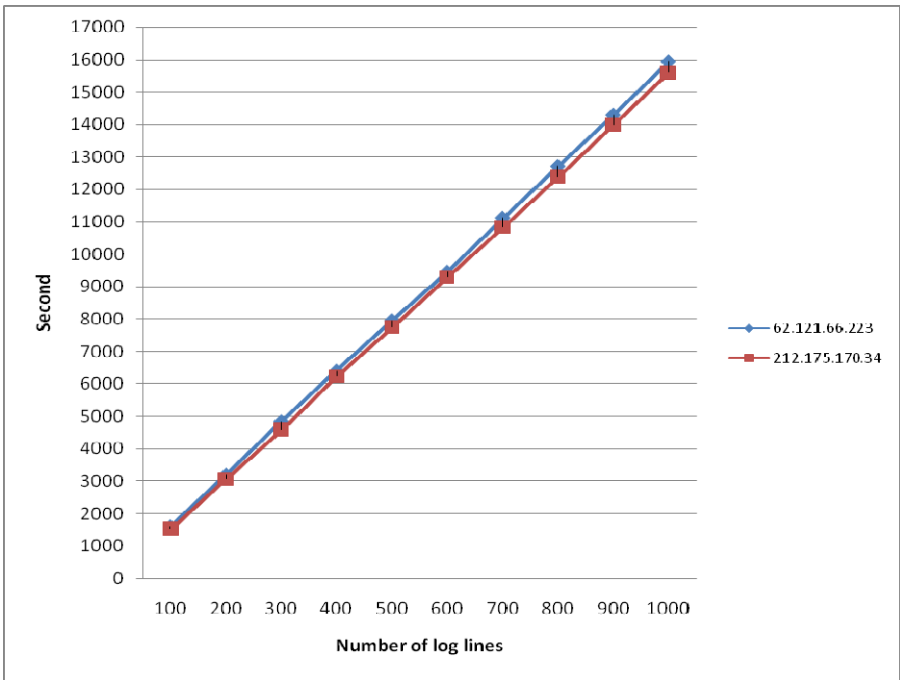


Figure 19 : Total search times for Waters et al non index based search

When figures are joined together, the difference between total search times of two schemes can be seen more obviously. Total search time for the **62.121.66.223** keyword can reduce about 21 times and for the **212.175.170.34** keyword which is more frequent can reduce 72 times in the index based scheme.

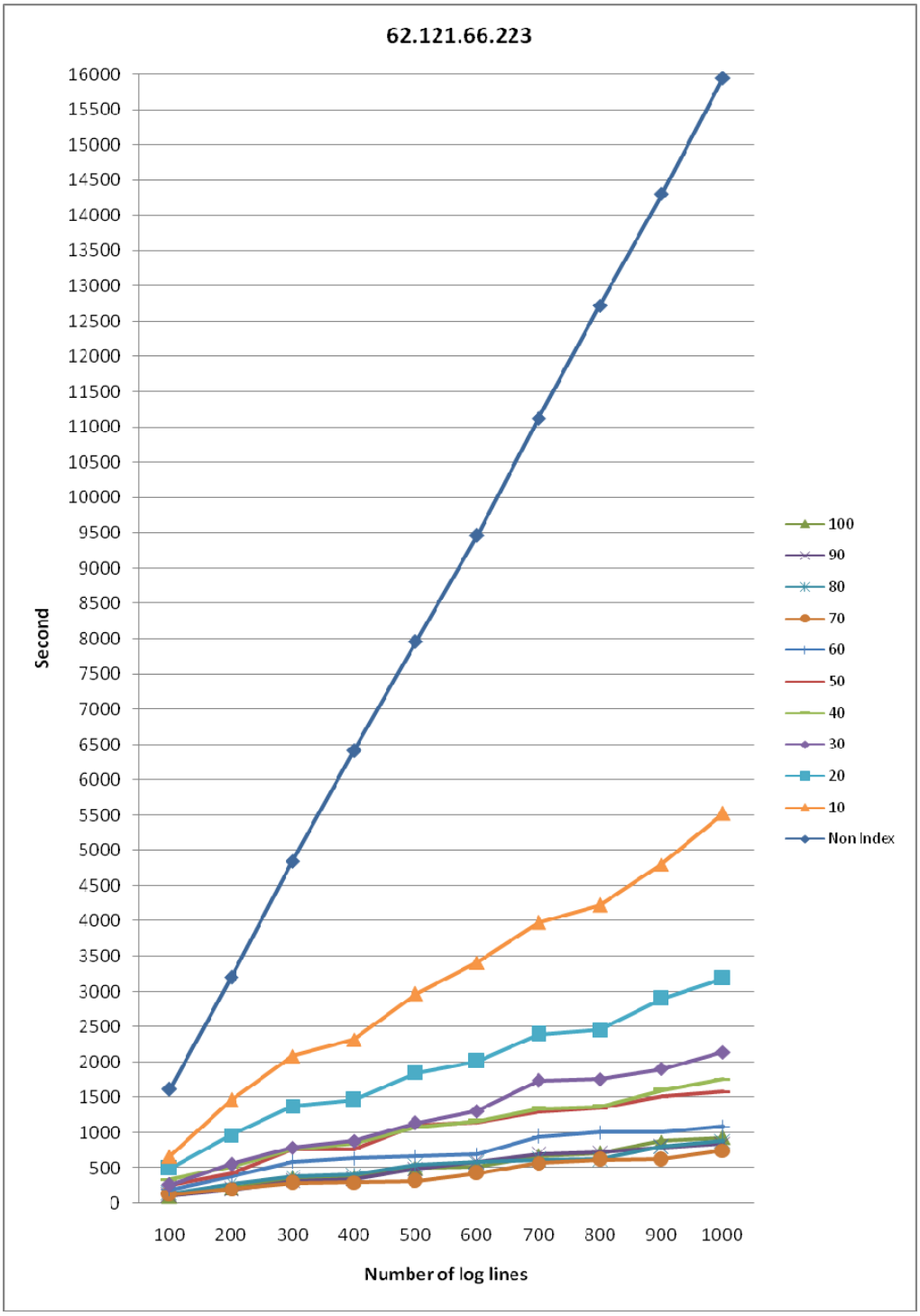


Figure 20 : Total search times for Waters et al index based and non index based search using less frequent keyword

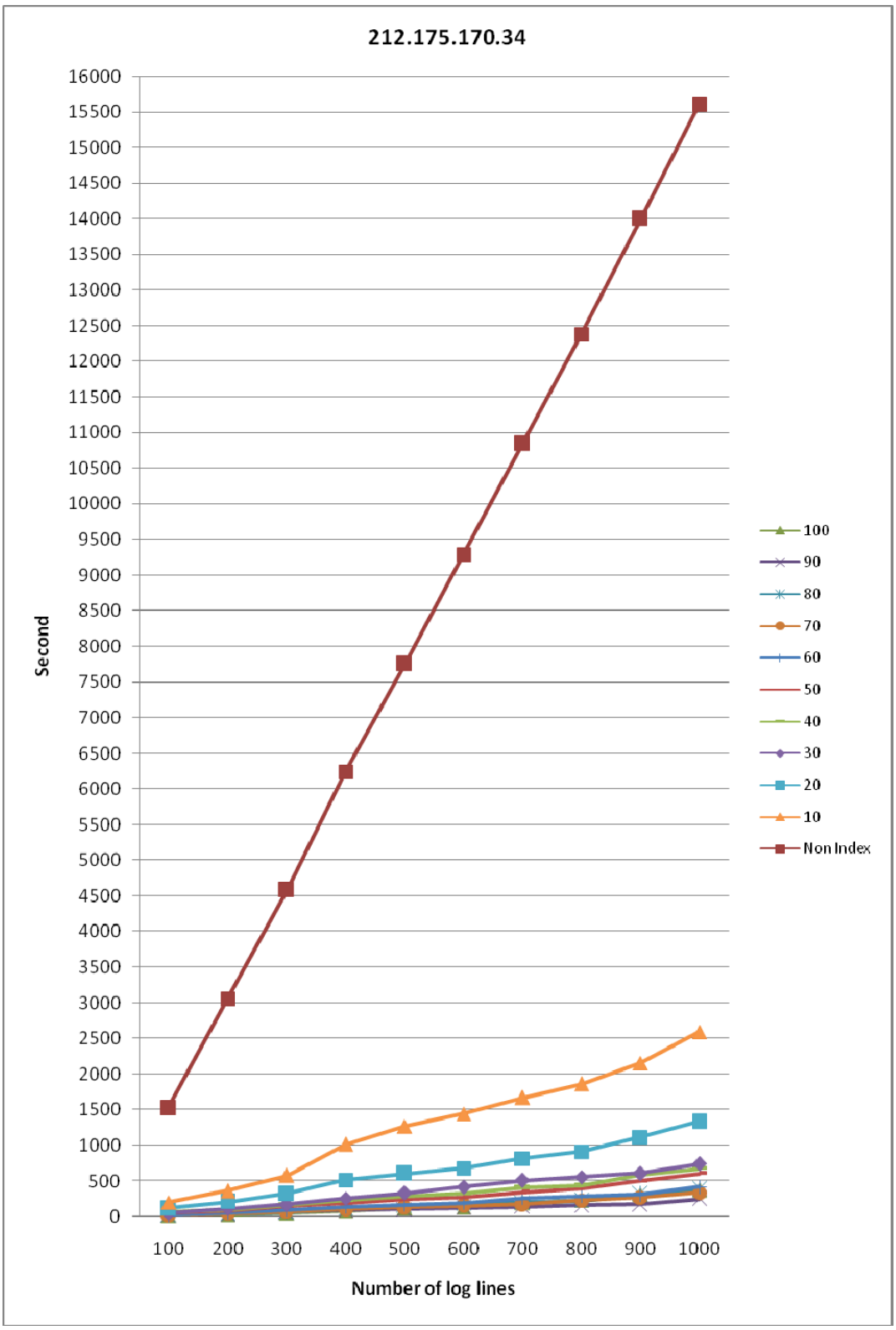


Figure 21: Total search times for Waters et al index based and non index based search using frequently repeated keyword

Also using same logs, we implement our scheme. Total search times of our proposed solution are shown below table and figure. Although total search times of our scheme are mostly higher than those indexed based solution of Waters et al., it is not as high as non index based solution.

Table 4: Total search times for our proposed solution in different tests

	62.121.66.223	**212.175.170.34**
Test 1	4850544	4729099
Test 2	4755718	4710004
Test 3	4787702	4648281
Test 4	4924661	4725468
Test 5	4830000	4717065
Average	**4829725**	**4705983.4**

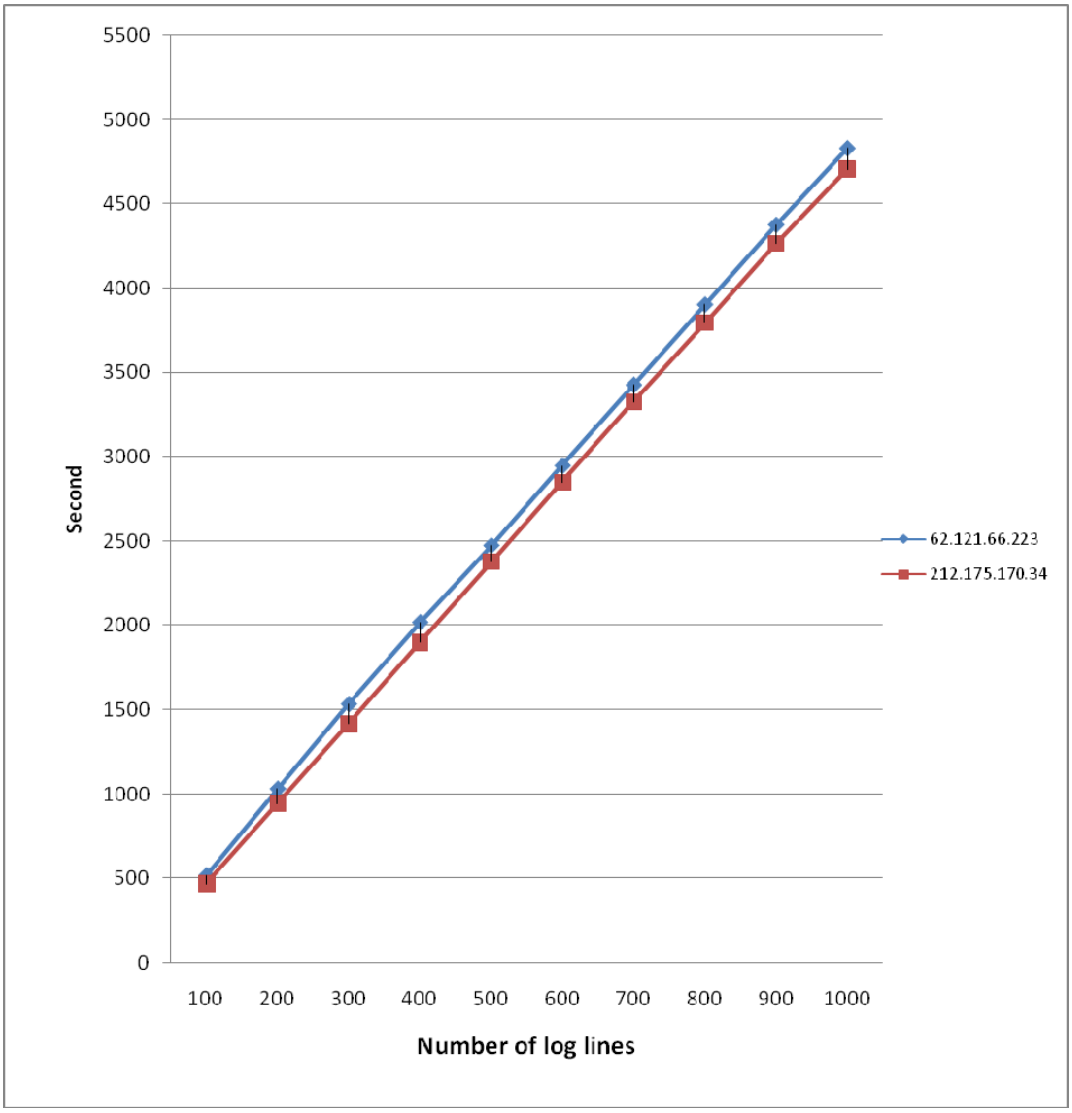


Figure 22 : Total search times for our proposed column based search

Total search times for the Indexed, Non Indexed schemes of Waters et al and our column based scheme are shown below.

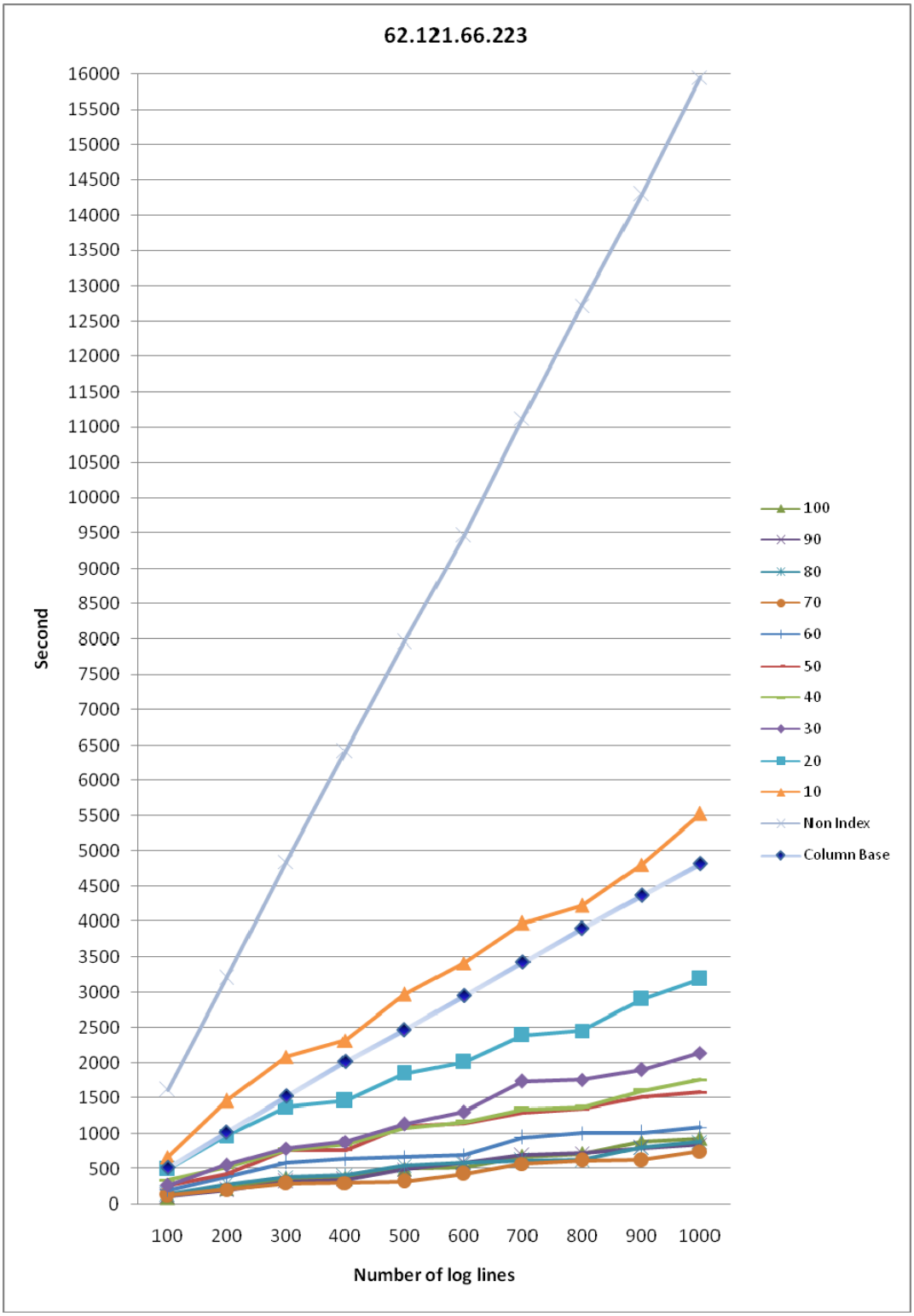


Figure 23 : Total search times for the Indexed, Non Indexed schemes of Waters et al and our column based scheme using less frequently repeated keyword

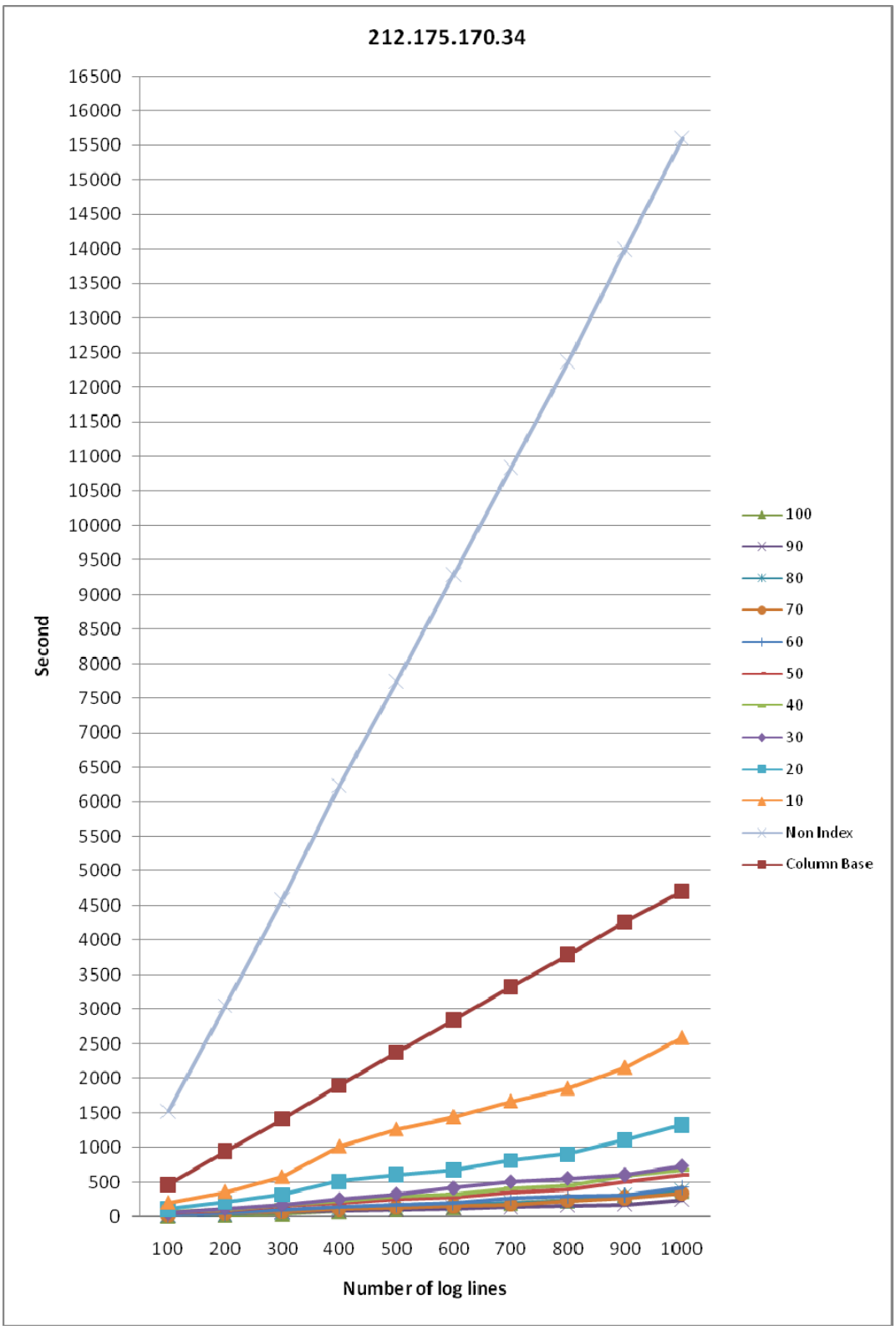


Figure 24 : Total search times for the Indexed, Non Indexed schemes of Waters et al and our column based scheme using frequently repeated keyword

Above two figures show that although index based scheme uses single keyword search, it has a higher performance than our column based scheme and non index based scheme of Waters et al.

As Waters et al, we design and implement an index based solution which is based on our column based approach and "block" idea of Waters et al. There are two main reasons using blocked index method rather than fully index (non blocked) method:

1- In the non blocked index method, it is needed to process each log entry line by line. When a log record is produced by the system, keywords from this log entry are required to be extracted and located to their index. This brings high computation overhead because for each keyword, actually it is needed make a search operation in the production phase.

2- When a log record is produced by the system, keywords from this log entry are required to be extracted and located to their index. While extracted keywords are located to their index, an attacker who gains access to logging machine can learn the index of that keyword by monitoring the logging process.

By using block idea, logs are generated as follows in a block. Let's assume that a block contains "*t*" lines of log:

1. Random symmetric encryption keys, $K_1 \ldots K_t$, are created for one time use.

2. Each log entry is encrypted using K_i

$$E_{\mathbf{K_i}}(R_\mathbf{i})$$

3. For each column (i.e. Source IP column, Destination IP column) distinct keywords (**w)** and their indices are extracted. N shows the number of same keyword with in a column, “i” represents the indices and “j” represents the distinct keyword in that column.

$\mathbf{w}_{SourceIP,j} \rightarrow \{i_{SourceIP,j,1}, \ldots, i_{SourceIP,j,N}\}$

.

.

$\mathbf{w}_{DestIP,j} \rightarrow \{i_{DestIP,j,1}, \ldots, i_{DestIP,j,N}\}$

4. For each keyword, the Identity Based Encryption Public Key is computed using public parameters of trusted third party and access permission for this log type.

$ibePK_{SourceIP,j} = (T_j \mid \mathbf{w}_{SourceIP,j})$

.

.

$ibePK_{DestIP,j} = (T_j \mid \mathbf{w}_{DestIP,j})$

5. Using Identity Based public keys for each keyword, keyword indices concatenated with random symmetric keys for each column are encrypted as follows:

$\mathbf{c}_{SourceIP,j} = IBE_{\mathbf{ibePK}_{SourceIP,j}}(\mathbf{w}_{SourceIP,j} \mid i_{SourceIP,j,1} \mid K._{i_{SourceIP,j,1}} \mid \ldots \mid i_{SourceIP,j,N} \mid K._{i_{SourceIP,j,N}})$

.

.

$\mathbf{c}_{DestIP,j} = IBE_{\mathbf{ibePK}_{DestIP,j}}(\mathbf{w}_{DestIP,j} \mid i_{DestIP,j,1} \mid K._{i_{DestIP,j,1}} \mid \ldots \mid i_{DestIP,j,N} \mid K._{i_{DestIP,j,N}})$

6. Indexed log block that contains encrypted logs and index are written to the log file as follows:

$\mathbf{c}_{SourceIP,1}$, …. $\mathbf{c}_{SourceIP,M}$

.

.

$\mathbf{c}_{DestIP,1}$, …. $\mathbf{c}_{DestIP,M}$

$E_{\mathbf{K_1}}(R_1)$

.

.

$E_{\mathbf{K_t}}(R_t)$

Search operation is made by an investigator as follows: Let's assume that investigator wants to search for source IP address **62.121.66.223.** First, he gets capability for that IP address. Capability will be calculated using ($T_{\mathbf{InternetAccessLogs}}$ | IP)

Suppose that we get a capability for that source IP address in Internet access logs. Then search operation will be made source IP part of each block as follows:

$\mathbf{c}_{SourceIP,1}$ is decrypted using capability. If result contains that searched source IP address, symmetric keys of encrypted logs and indices are extracted. Encrypted logs which are pointed by the indices are decrypted using these symmetric keys for that block. Then, following source IP part of other blocks is tested.

Again we also implement this scheme on the above problem setting: we make a search operation using destination IP number **212.175.170.34** which is frequently repeated among log entries and source IP number **62.121.66.223** which is less frequent when compared to the other one. Again we also increase block size to show how block size effect total search time. Below figures show the results. Our proposed new index based scheme has the highest performance of all others.

Table 5 : Total search times for 62.121.66.223 in different block sizes and tests

	62.121.66.223									
	Block size 10	Block size 20	Block size 30	Block size 40	Block size 50	Block size 60	Block size 70	Block size 80	Block size 90	Block size 100
Test 1(sec)	1848091	1094890	707067	617737	562718	458651	247205	310025	288520	306124
Test 2(sec)	1851746	1098481	705746	606967	545390	382162	247620	313268	288085	306812
Test 3(sec)	1828112	1098704	716306	615483	550763	381367	243716	311719	287100	296422
Test 4(sec)	1830679	1085878	702762	611560	558330	386729	246759	320180	283362	306721
Test 5(sec)	1824294	1092966	695768	607764	573174	383878	245403	316892	287827	302827
Average(sec)	1836584	1094184	705530	611902	558075	398557	246141	314416	286979	303781

Table 6 : Total search times for 212.175.170.34 in different block sizes and tests

	212.175.170.34									
	Block size 10	Block size 20	Block size 30	Block size 40	Block size 50	Block size 60	Block size 70	Block size 80	Block size 90	Block size 100
Test 1(sec)	732960	382032	224072	222220	179659	120985	110611	121891	77704	103390
Test 2(sec)	737189	377521	223000	211940	175656	120460	110363	114734	77463	104204
Test 3(sec)	745409	381810	247790	210267	172984	119782	109820	115586	77137	103064
Test 4(sec)	741673	381670	222239	211798	181281	124343	112121	117804	78193	104986
Test 5(sec)	740560	382644	221410	212998	172641	116301	111729	118310	76609	102217
Average(sec)	739558	381135	227702	213845	176444	120374	110929	117665	77421	103572

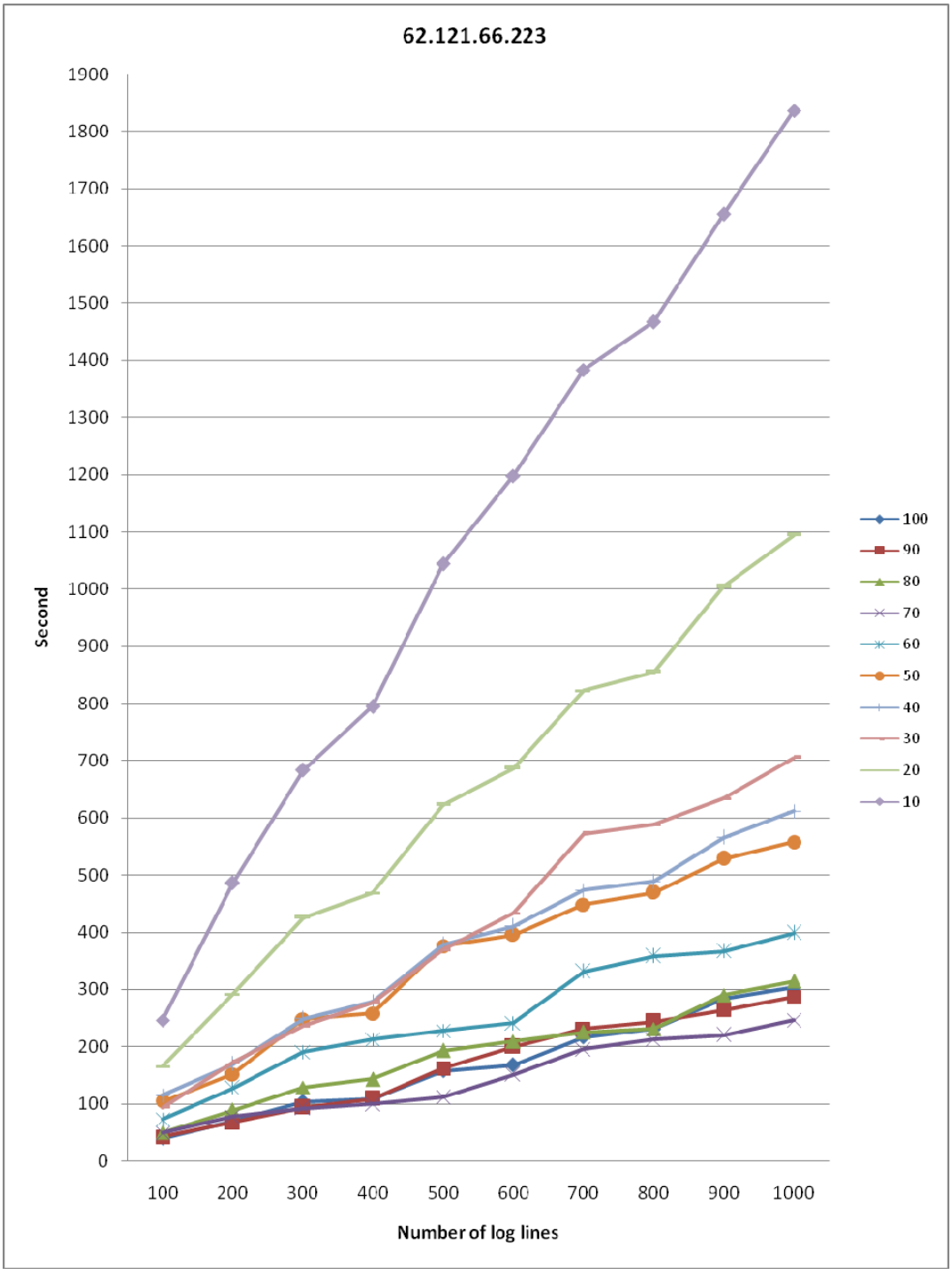


Figure 25 : Total search times for our index based scheme using less frequently repeated keyword

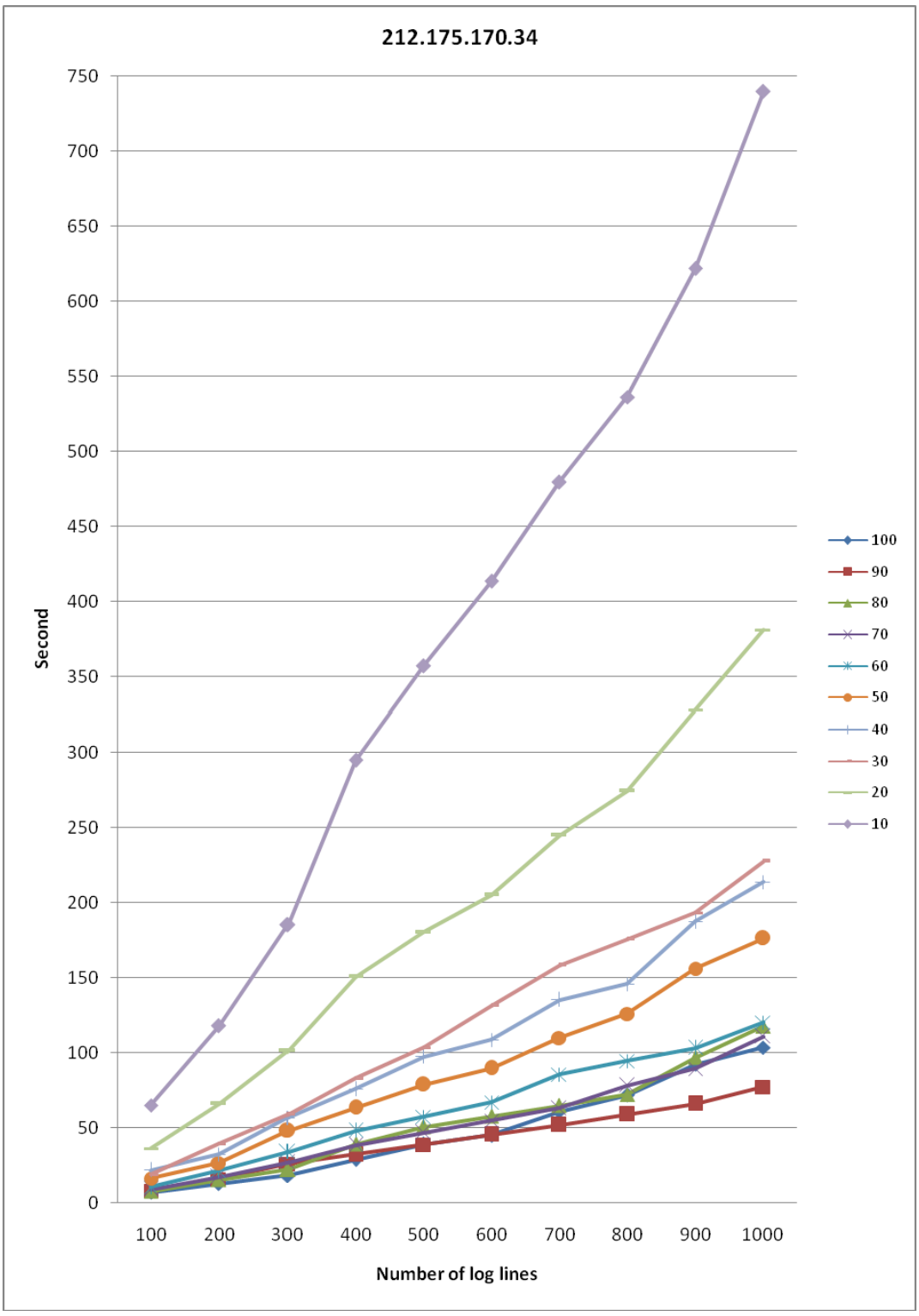


Figure 26 : Total search times for our index based scheme using frequently repeated keyword

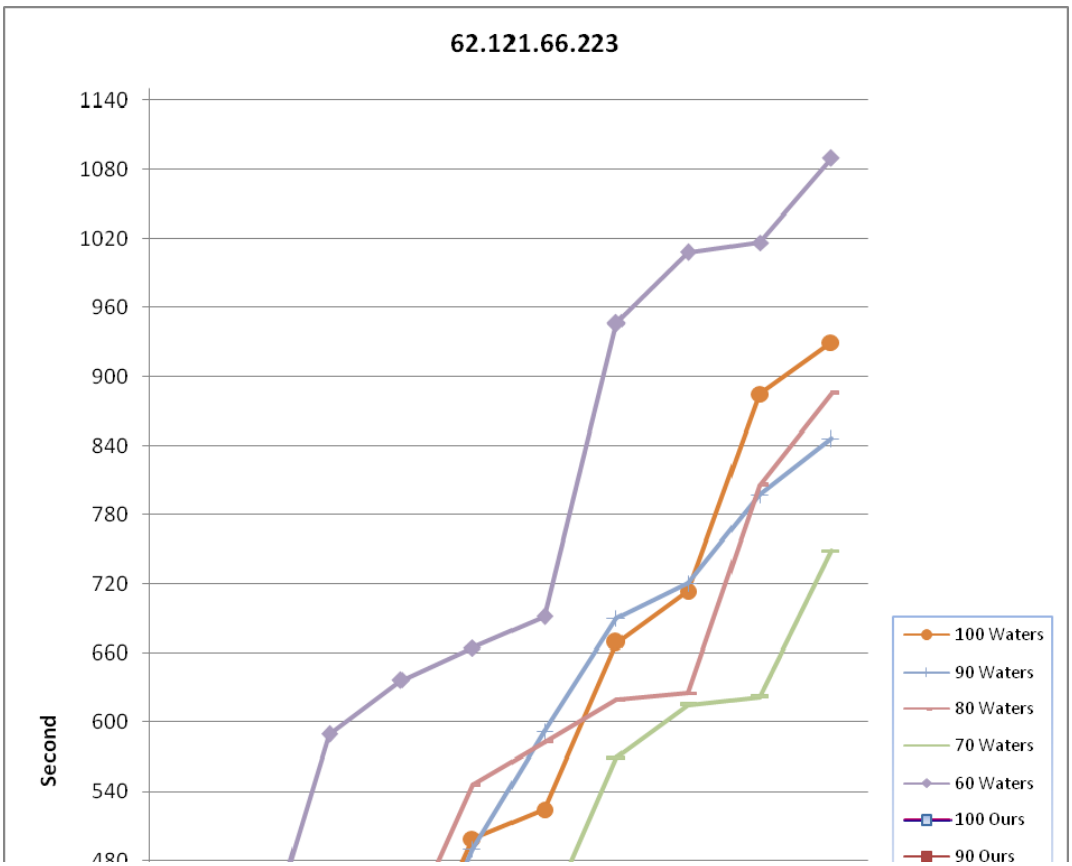


Figure 27 : Total search times for the index based scheme of Waters et al and our index based scheme using less frequently repeated keyword for from 60 to 100 blocks

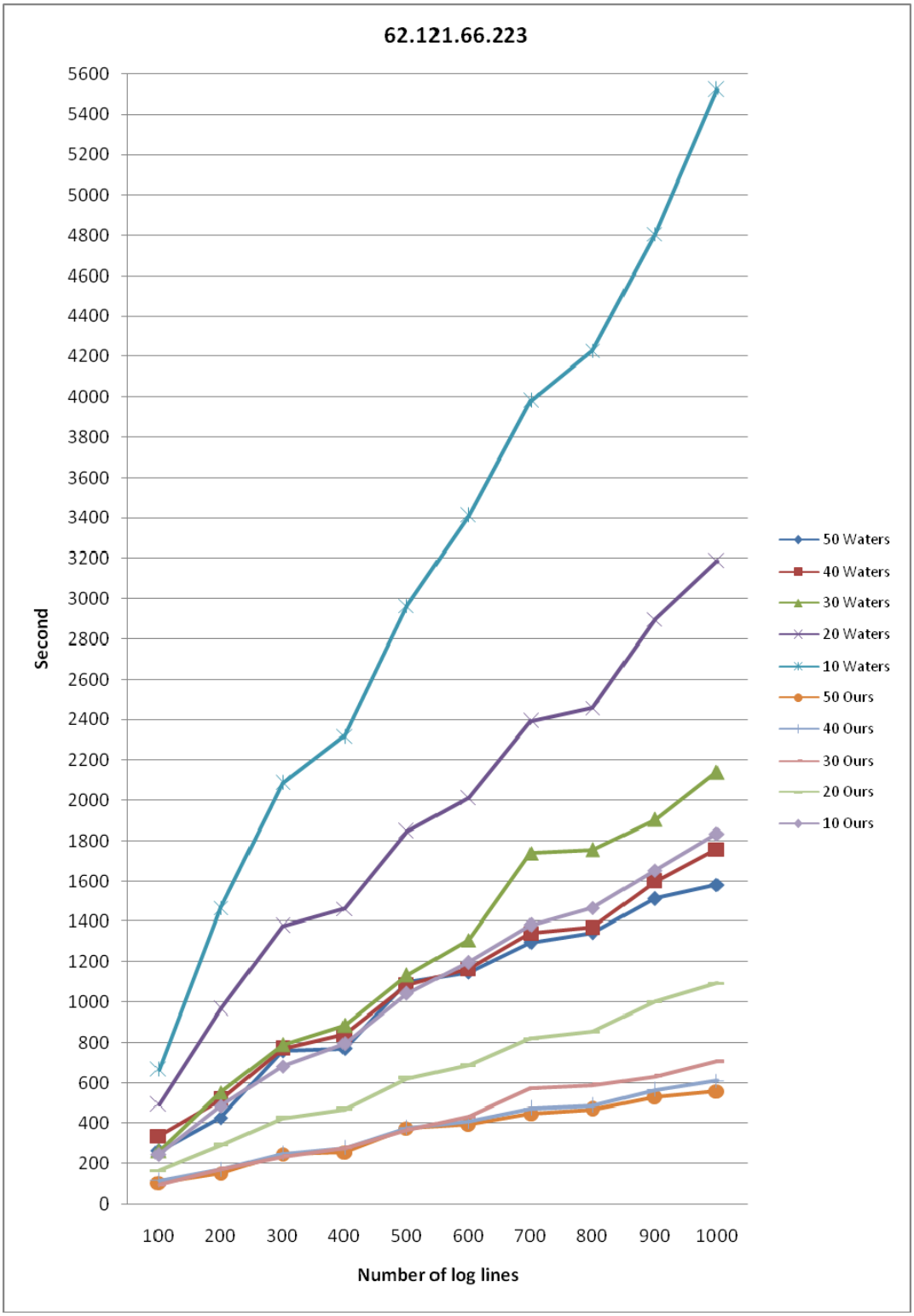


Figure 28 : Total search times for the index based scheme of Waters et al and our index based scheme using less frequently repeated keyword for from 10 to 50 blocks

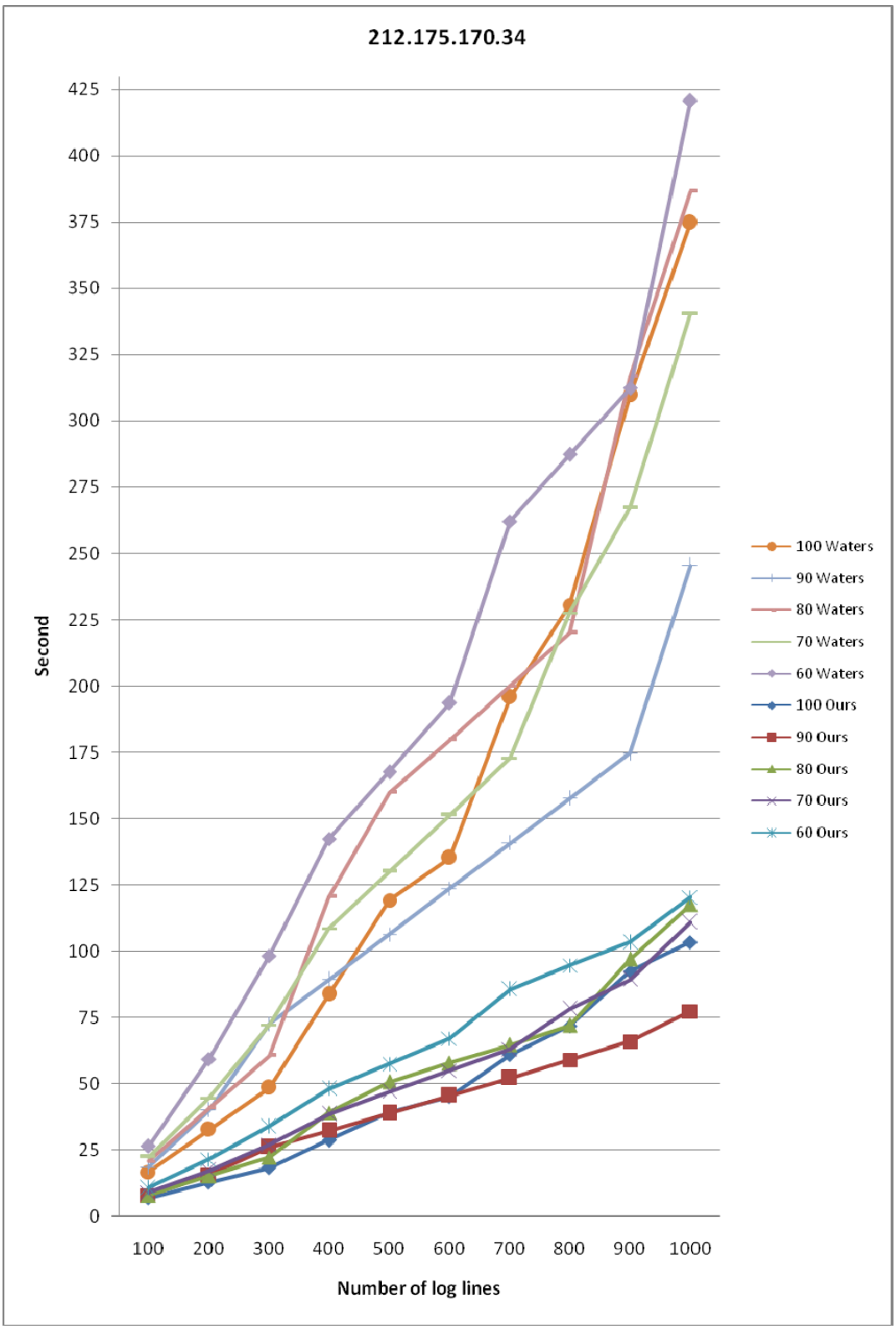


Figure 29 : Total search times for the index based scheme of Waters et al and our index based scheme using frequently repeated keyword for from 60 to 100 blocks

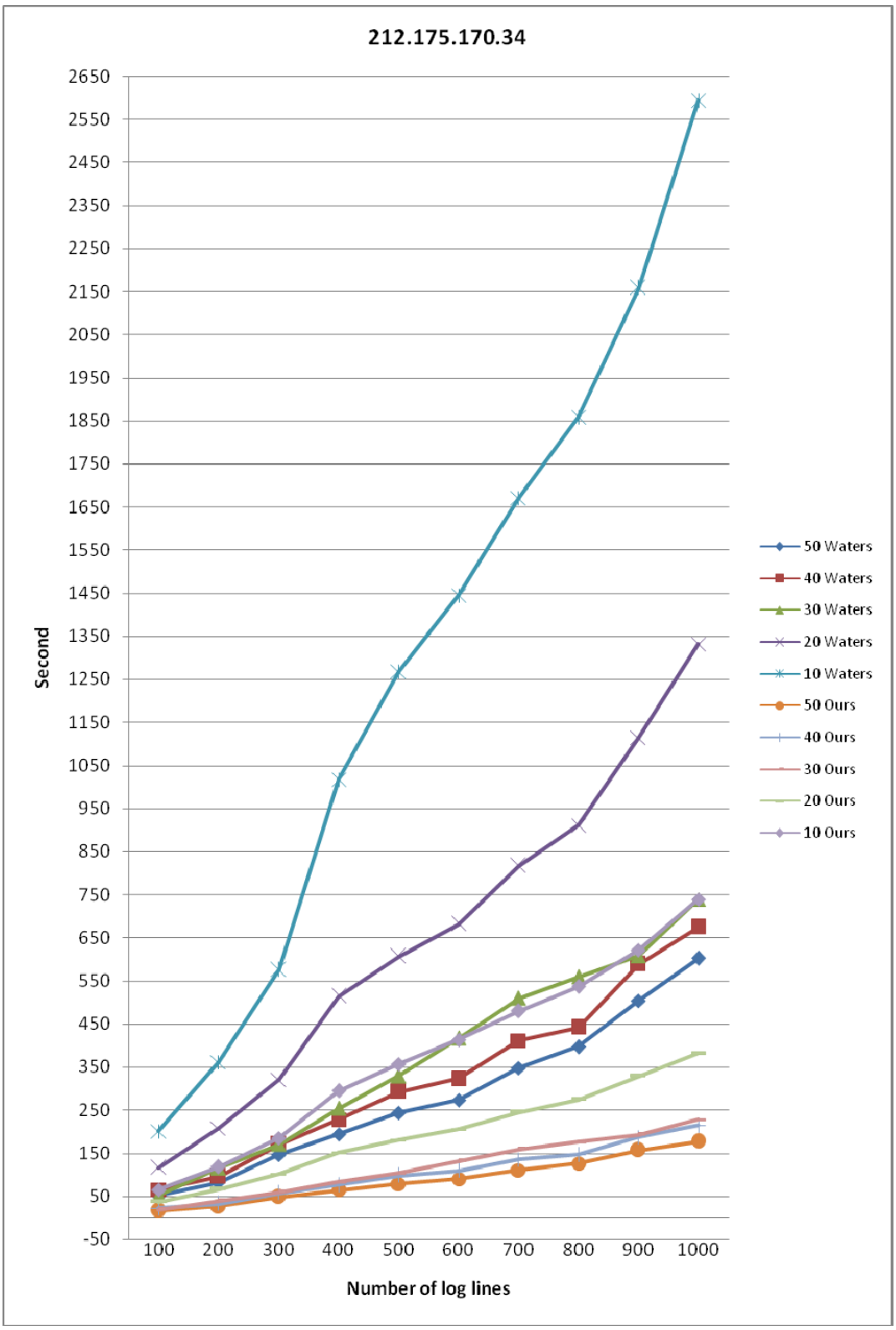


Figure 30 : Total search times for the index based scheme of Waters et al and our index based scheme using frequently repeated keyword for from 10 to 50 blocks

IV.3.6 Our Contributions

Waters et al. proposed an index based solution but they did not implement it. In this thesis, to compare methods of Waters et al. with ours, we implement not only non index based method and but also index based method of Waters et al. Also we design and implement an index based solution which is based on our column based approach.

As we stated before studies searching on secure audit log focused on the single keyword search which often yields far too coarse results. Our main contribution in this thesis is the proposition and implementation of column base approach to encrypted logs by using logs' predetermined structure, while searching, this approach enables us define multiple criteria that helps analyzing logs. In other words, our mechanism enables an investigator search encrypted logs making different combinations of keyword.

Verification of logs should be thought in two parts: 1- To detect any alteration, deletion and insertion, individual log records should be linked together. 2- If any alteration, deletion is conducted on records, to be able to make the rest of the records useful, it has to be provable that the rest of the logs are still authentic. In other words, each log record must be verifiable. [36]

In our solution, since forward integrity property is supported, we are able to detect any alteration, deletion and insertion on logs in an effective manner. When related studies are examined, it can be seen that secure audit logging schemes are based on Schneier and Kelsey's mechanism which does not provide single log verification. The main problem of the Schneier and Kelsey mechanism is that they depend on the previous values to detect any anomaly but when one log is deleted from the chain, it is not possible to verify the next log whether it is altered or not. This causes whole log next after the deleted one to be doubtful in terms of its authenticity. In other words, this scheme does not provide single log verification.

To handle this, in our scheme we provide single log verification, by this way; we prevent any doubt about the logs next after the deleted one.

Like other methods, our model provides secure delegation of search capabilities to authorized users while protecting information privacy but also our method sets boundaries of a user's search operation. This idea is not defined before for a search operation. In our model, when an investigator wants to search logs, he sends some keywords to the Trusted Third Party to get search capabilities. According to access permission of the investigator, capabilities are constructed. Therefore, even when the user can reach all the log set; he can not obtain information which is not related with his case.

IV.3.7 System

One centralized point of storage for log is easier to secure, easier to backup, easier to acquire for analysis. Therefore, logs from different resources should be directed to this centralized server and most network devices such as routers, switches, firewalls, and other servers have capability to send their log file to a centralized point. Storing secure audit logs on a secured remote server constitutes important part of our logging infrastructure and enables us to control accesses to this central server.

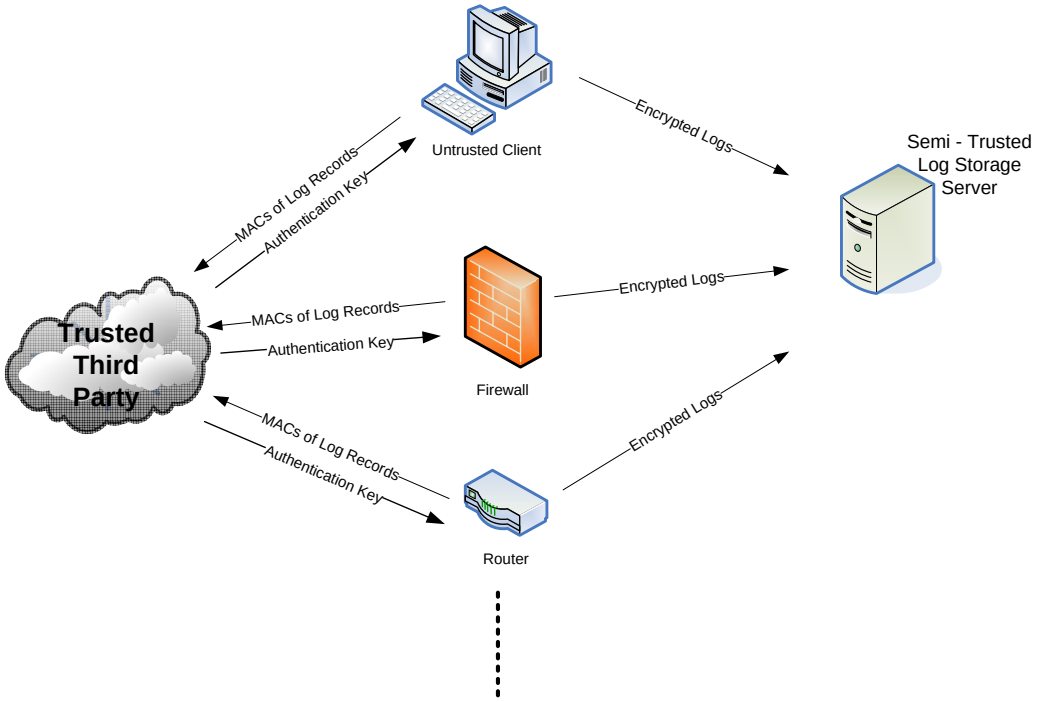


Figure 31 : Encryption of logs

When an untrusted client or any other type of network device starts logging, first it gets authentication key from trusted third party. After it gets this initial value, it starts logging and after a predetermined time, sends secure audit logs to central semi-trusted log storage server. It is a matter of policy in which intervals logs are sent to the semi-trusted log storage server. MACs of log records are sent to Trusted Third Party.

When an investigator or an analyzer wants to analyze or search secure audit logs, first, he has to authenticate himself to the Trusted Third Party. After the authentication, he sends keyword, or keywords that are going to be searched in the secure audit logs. Trusted Third Party evaluates the investigator request and gives capabilities according to his access rights or if he has not got a permission to search, simply denies his request. By using these capabilities, investigator searches secure audit logs

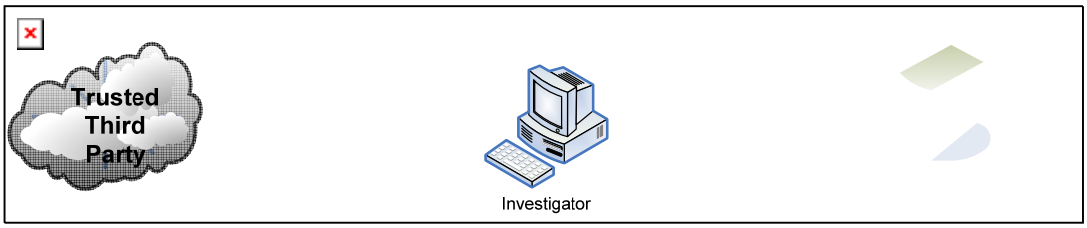


Figure 32 : Analyzing encrypted logs

IV.3.8 Discussion of the system

In our system, we separate log verification, log retention and decryption operation. By this way, strong security is provided for the audit logs. Integrity of logs is controlled, whenever needed by the trusted third party. By this way, firstly, when encrypted logs are received by the semi-trusted log storage server, trusted third party can verify whether any modification happens or not on the logs by using initial authentication key. Also, semi-trusted log storage server can check the integrity of the logs by using hash chain. Secondly, integrity of logs is protected against malicious administrators. Changes by an administrator of semi-trusted log storage server on encrypted logs can easily be detected by trusted third party and this prevents or mitigates the potential damage from stealthy threat.

By using public key cryptography, we divide the encryption and decryption operations. By this way, on untrusted machine, encryption operation is conducted without revealing any past information about logs even an attacker gains access to the server and although semi-trusted log storage server does not know the content of logs, an investigator can search any information at semi-trusted log storage server by using encrypted logs.

A logging system can encounter different types of attacks. They can be classified in the following categories [48]:

Read: An attacker can gain knowledge like vulnerabilities of the system or other confidential information by using content of log.

Write: By modifying or inserting false log records, attacker can mislead forensic investigators or administrators.

Delete: To cover up their traces, attacker tries to remove log records which show presence of them.

Denial of Service: to prevent logging, attacker can carry out DoS attack.

Flooding: Attackers try to conceal log records that can reveal their attack, in this type of attack, attackers conduct their attack and let the logging system to log them but at the same time they send thousands of valid requests. By this way they bury their traces very deep in the log ocean. This attack can be best defined with the problem of “finding a needle in a haystack". [49]

Abuse of trust: A trusted administrator can abuse their existing privileges and may harm the logging system.

Extraordinary Events: System failures or accidental incidents may harm logging system.

Previous studies which focused on security of audit logs provide satisfactory solutions against read/write/delete attacks but for denial of service, flooding and abuse of trust attacks. Our system can help to find some solutions these unaddressed problems.

A denial of service attack is an attack in which resources of a system is exhausted in a way that no one can benefit from the service. DoS attacks can easily be detected but hard to counter. The aim of an attack to our system in terms of DoS can be attempted to prevent communication between secure logging machine and

trusted servers with initial keys. To avoid such an attack, secure communication channel between secure logging machine and trusted servers can help us.

It is hard to prevent log flooding attack, but a good countermeasure against log flooding attack is detection of this malicious activity. In the previous mechanisms, it is hard to detect such an attack because they are mostly based on a single keyword search. By using our solution, an investigator can make different combinations of keywords and can compare results with results coming from logs of other machines. Malicious activity by this way can easily be detected when compared to the other secure audit logging solutions.

According to survey in [50] it is state that "In 57% of the cases, the insiders exploited or attempted to exploit systemic vulnerabilities in applications, processes, and/or procedures (e.g., business rule checks, authorized overrides)". Since insiders have full knowledge on the system, it is hard to prevent them from abuses their privileges. To handle this, we provide distributed security, in other words we distribute trust between trusted third party, semi-trusted log storage server and investigator.

CHAPTER V

IMPLEMENTATION OF OUR PROPOSED SOLUTION ON IPFILTER FIREWALL

We have implemented our scheme using logs of IPFilter firewall which is called IPMon. As IPMon is a part of IPFilter firewall, we design an Intelligent Security Management System (I-SMS) to govern IPFilter firewall and monitor network access over encrypted logs. We assume that trusted third party in our scheme can also take the role of searching and validating logs for the sake of illustration of our idea.

To implement our secure analyzable audit logs, we choose java as the programming language and Borland JBuilder X and Eclipse as a development environment. The hard part on the implementation of our mechanism in java is using IBE. Therefore, we get help from [51] which has a Java Cryptographic Architecture (JCA) integrated implementation of IBE. For other cryptographic operations such as MAC and Symmetric Encryption, we use Bouncy Castle Provider [52].

The Unified Modeling Language (UML) is a graphical notation for drawing diagrams of software concepts. UML can be used for drawing diagrams of a problem domain, a proposed software design, or already completed software implementation [53]. We demonstrate our java classes using UML. For all program we write about 30.000 lines of code.

Specifically an IBE scheme mostly composed of four sections:

(1) **Setup** generates global system parameters and a master-key,
(2) **Extract** uses the master-key to generate the private key corresponding to an arbitrary public key string ID (identity),
(3) **Encrypt** encrypts messages using the public key ID,
(4) **Decrypt** decrypts messages using the corresponding private key for public key ID.

Setup, Extract and Decrypt operations are realized on the trusted third party and Encrypt operations are realized on the IPFilter firewall while generating logs encrypted with AES (Advanced Encryption Standard) symmetric cipher.

V.1 Operations on Firewall Host

Saying network security starts with the firewall is not wrong. As whole communication of network can be managed by firewall, our IPFilter firewall host helps us to control and monitor the whole network and hide internal network topology from attackers but communicating with this host requires some type of protocol as we implemented below:

V.1.1 Protocol Structure

Our communication protocol consists of three parts:

1- Codes expected from trusted third party:

When our trusted third party communicates with this IPFilter firewall host, it has to send some codes to be understandable from this host:

Table 7 : Codes expected from trusted third party

201="KULL"	201="USER"
202="PARO"	202="PASS"
203="GETIR"	203="RETR"
204="KAYDET"	204="STOR"
205="ANAHTAR"	205="KEY"
206="CIK"	206="QUIT"

2- Codes expected from IPFilter firewall host:

IPFilter firewall host sends message codes to trusted third party after each correct request to notify trusted third party about next step of the protocol:

Table 8 : Codes expected from IPFilter firewall host

101="I-SMS Firewall 1.0'a baglanildi"	101="Connected To I-SMS Firewall 1.0"
102="Lutfen parolanizi gonderiniz"	102="Password Required for"
103="Kullanici Dogrulandi"	103="User Authenticated"
104="Veri baglantisi kuruluyor, Port No"	104="Data Connection on Port"
105="Aktarim Tamamlandi"	105="Transfer Completed"
106 ="Seed Değerinin Aktarimi Tamamlandi"	106 ="Seed Values' Transfer Completed"
110="iyi Gunler"	110="Good Bye"

3- Codes when an error occurred for a reason are sent to trusted third party:

Table 9 : Error Codes

501="Dogrulama Hatasi"	501="Authentication Error"
502="Aktarim Hatasi"	502="Transfer Error"
503="Güvenlik Duvari tekrar baslatilamadi"	503="Firewall could not be restarted"
505="Bilinmeyen Komut"	505="Unknown Command"
700="Bilinmeyen Protokol Komutu"	700="Unknown Protocol Command"

A working example of protocol is like; after firewall started, it waits for connections from the clients to be configurable from remote sides. To authenticate an entity, it needs “USER” code in English and “KULL” code in Turkish. When the server gets this code along with the name of entity, it expect for password with the code of “PASS”/ “PARO.” After getting username and password, it controls them and if they are correct, firewall authenticates the entity. Then communicating entity may want a file with the code “RETR”/ “GETIR” from firewall or wants to store a file with the code “STOR”/ “KAYDET” to firewall. If the communicating entity wants to close the connection, it uses the “QUIT” / “CIK” codes (Figure 30).

Above communication protocol works on the background of our mechanism and provides better understanding of each communicating entity.

V.1.2 Generating Secure Audit Logs

IPFilter firewalls mostly produces logs named as IPMon and we try to make these log records secure in our implementation. A usual structure of IPMon logs records as follows:

```
Jan 3 13:19:55 gate ipmon[50]: 13:19:54.995753 em0 @0:78 b
158.193.254.40,6881 -> 212.175.170.34,8080 PR udp len 20 90 IN
```

```
Jan 3 13:19:55 gate ipmon[50]: 13:19:54.995753 em0 @0:78 b
158.193.254.40,6881 -> 212.175.170.34,8080 PR udp len 20 90 IN

Jan 3 13:19:55 gate ipmon[50]: 13:19:54.995753 em0 @0:78 b
158.193.254.40,6881 -> 212.175.170.34,8080 PR udp len 20 90 IN

Jan 3 13:19:55 gate ipmon[50]: 13:19:54.995753 em0 @0:78 b
158.193.254.40,6881 -> 212.175.170.34,8080 PR udp len 20 90 IN

Jan 3 13:19:55 gate ipmon[50]: 13:19:55.024987 em0 @0:77 b
71.240.244.46,63998 -> 212.175.170.34,17514 PR tcp len 20 48 -S IN

Jan 3 13:19:55 gate ipmon[50]: 13:19:55.053594 em0 @0:77 b
218.6.247.244,11067 -> 212.175.170.34,58908 PR tcp len 20 48 -S IN
```

To define each field in our implementation, we construct a data object named as ipmDTOnames (Figure 33).

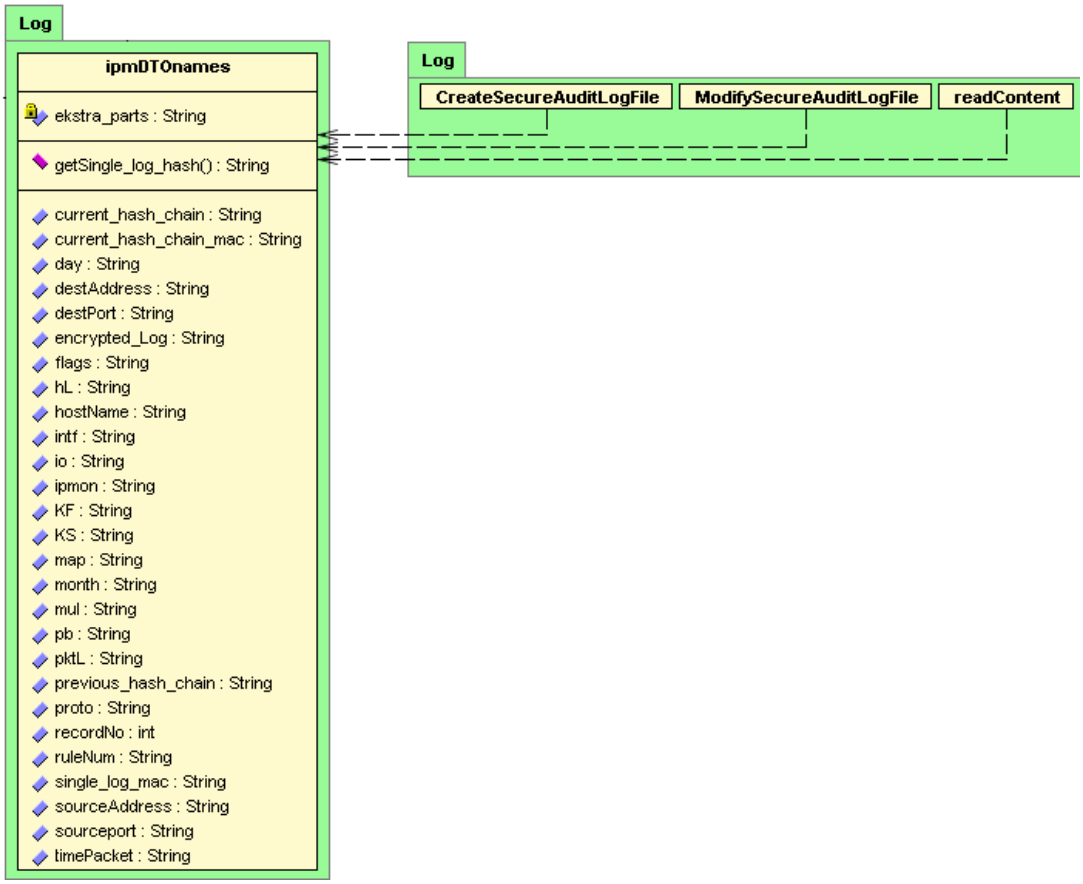


Figure 33 : Field Names DTO

To be able to use each field in our IBE scheme as a public key, there is a need for an extraction mechanism for this set of keywords. Therefore, we have implemented an extraction class (Figure 34), while realizing IBE encryption operations and other cryptographic operations.

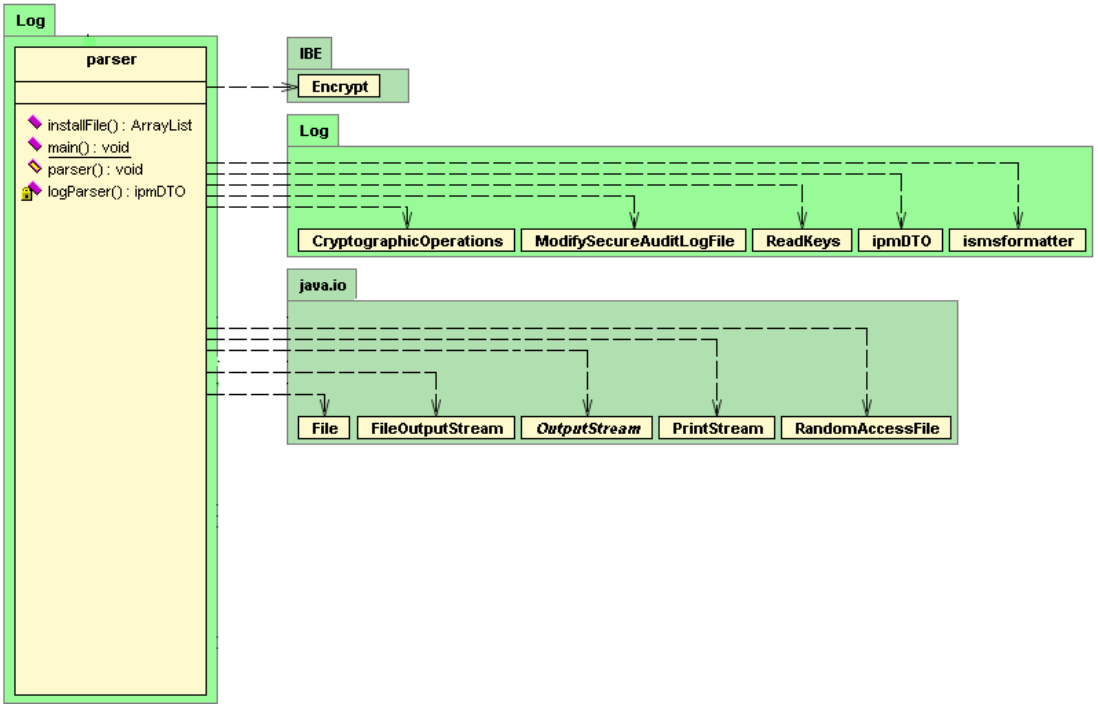


Figure 34 : Extraction Class

As we stated at the beginning, Firewall host uses Encrypt operation from IBE, random symmetric key to calculate symmetric encryption which provides confidentiality, hash and HMAC for validation of logs. Symmetric encryption, generation of random symmetric key, hash and HMAC are implemented in the CryptographicOperations class (Figure 35) and IBE Encrypt operation is implemented in the Encrypt class (Figure 36).

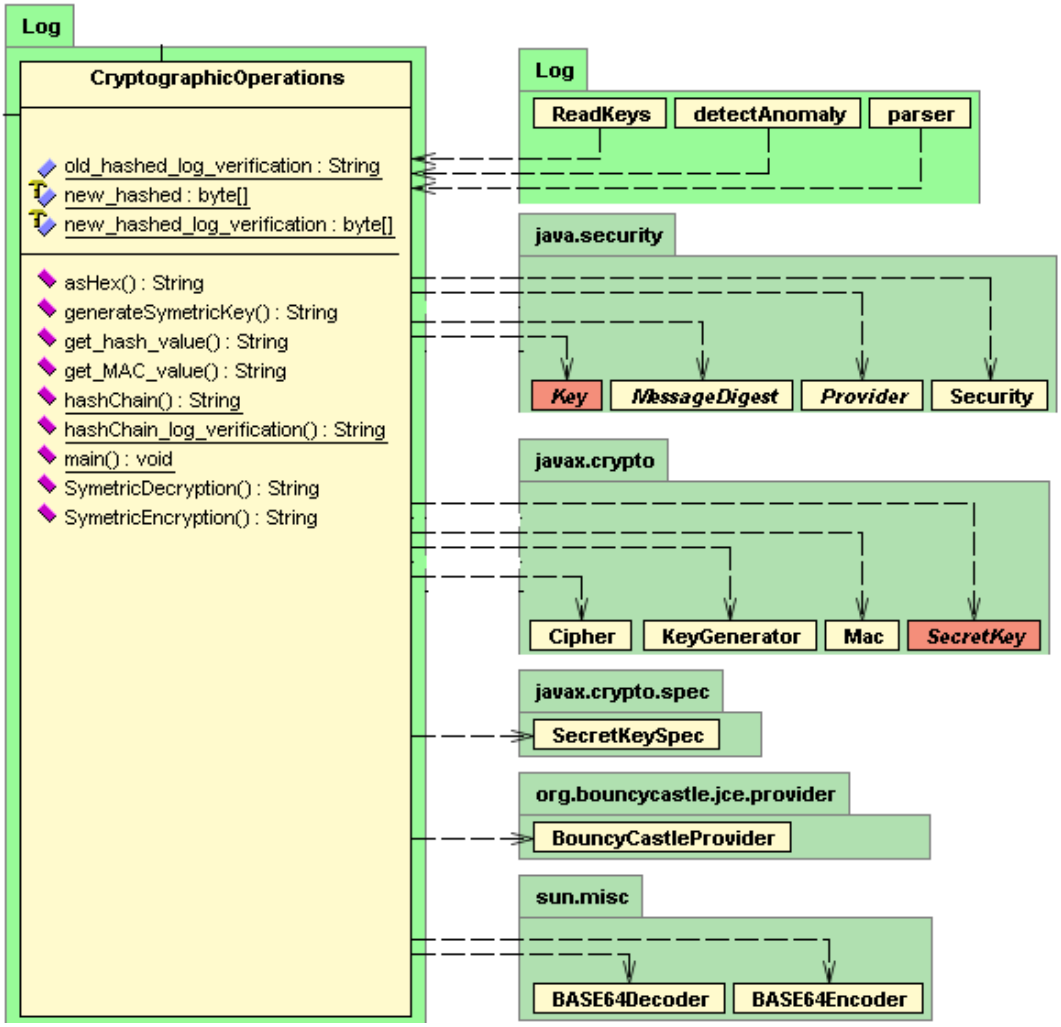


Figure 35 : Cryptographic Operations Class

Encryption: Uses *ID* as a public key and *params*, the sender encrypts plaintext message *M* and obtains a ciphertext *C*.

C = Encrypt (*params*, *ID*, *M*)

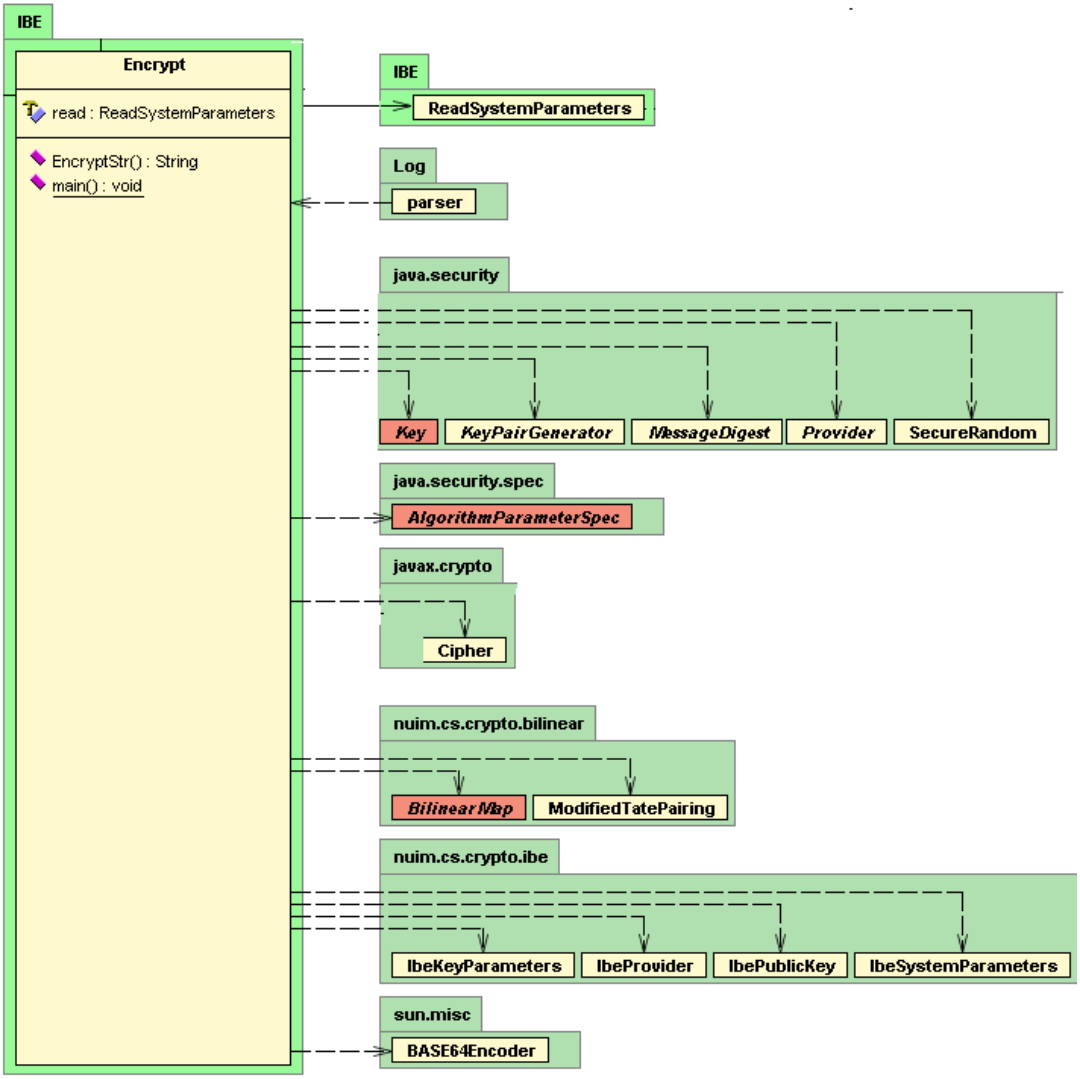


Figure 36 : IBE Encryption Class

V.2 Operations on Trusted Third Party

Trusted third party can manage firewall, send initial authentication key to firewall, and response search requests of investigators.

Setup, Extract and Decrypt (as a role of investigator) IBE operations are realized in the Trusted third party.

Setup: $k \in Z^{+}$, takes a security parameter k and generates *params* (system parameters) and *master key*. Since, master key is only known by Trusted third party, nobody except Trusted third party constructs Private Key (Figure 37).

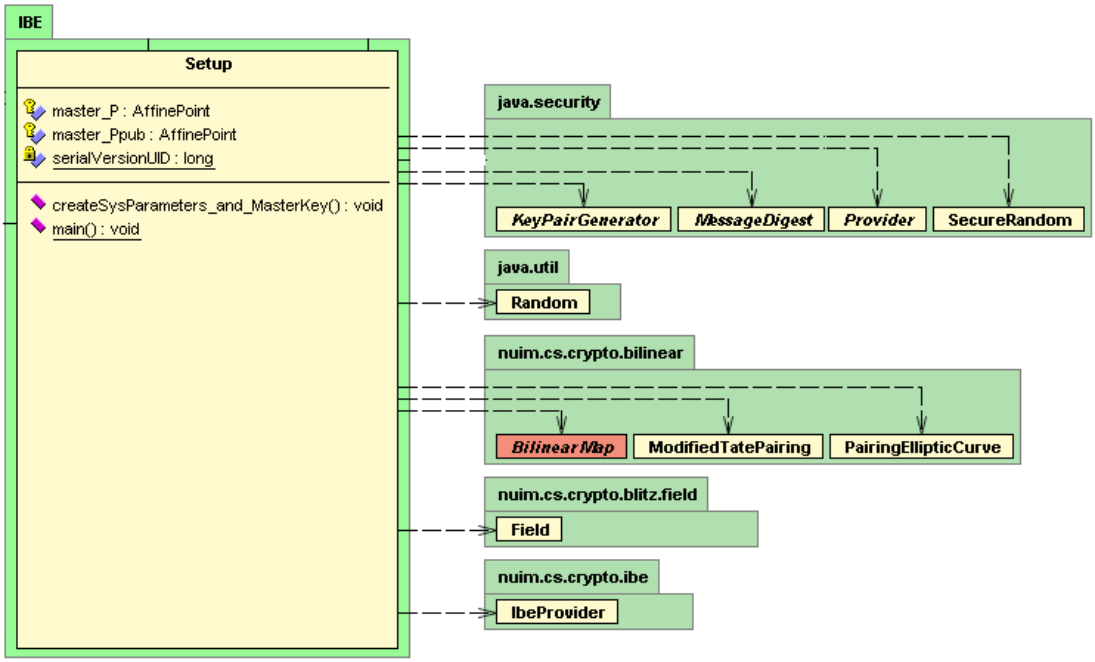


Figure 37 : IBE Setup

Private Key Extraction: Trusted third party uses the *master-key, params* to generate the private key corresponding to an arbitrary string *ID* such as e-mail address, phone number which is used as a public key (Figure 38).

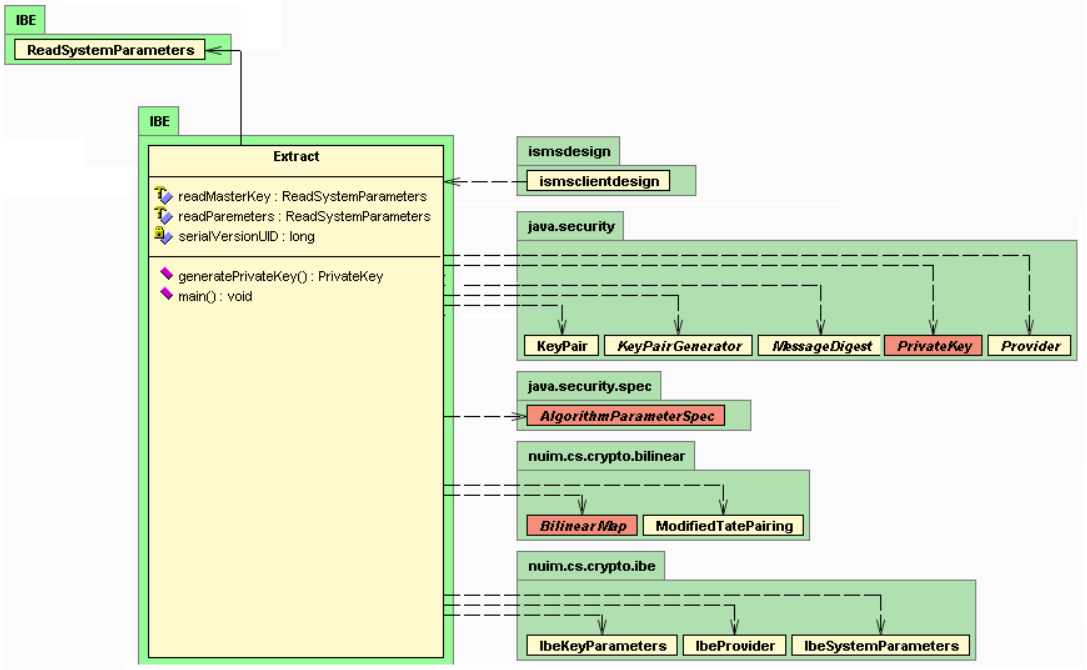


Figure 38 : IBE Extraction

Decryption: Takes as input ciphertext *C, params and private key* which is generated by the Private Key Extraction algorithm and returns plaintext message *M* (Figure 39).

M = Decrypt (*params*, *private key*, *C*)

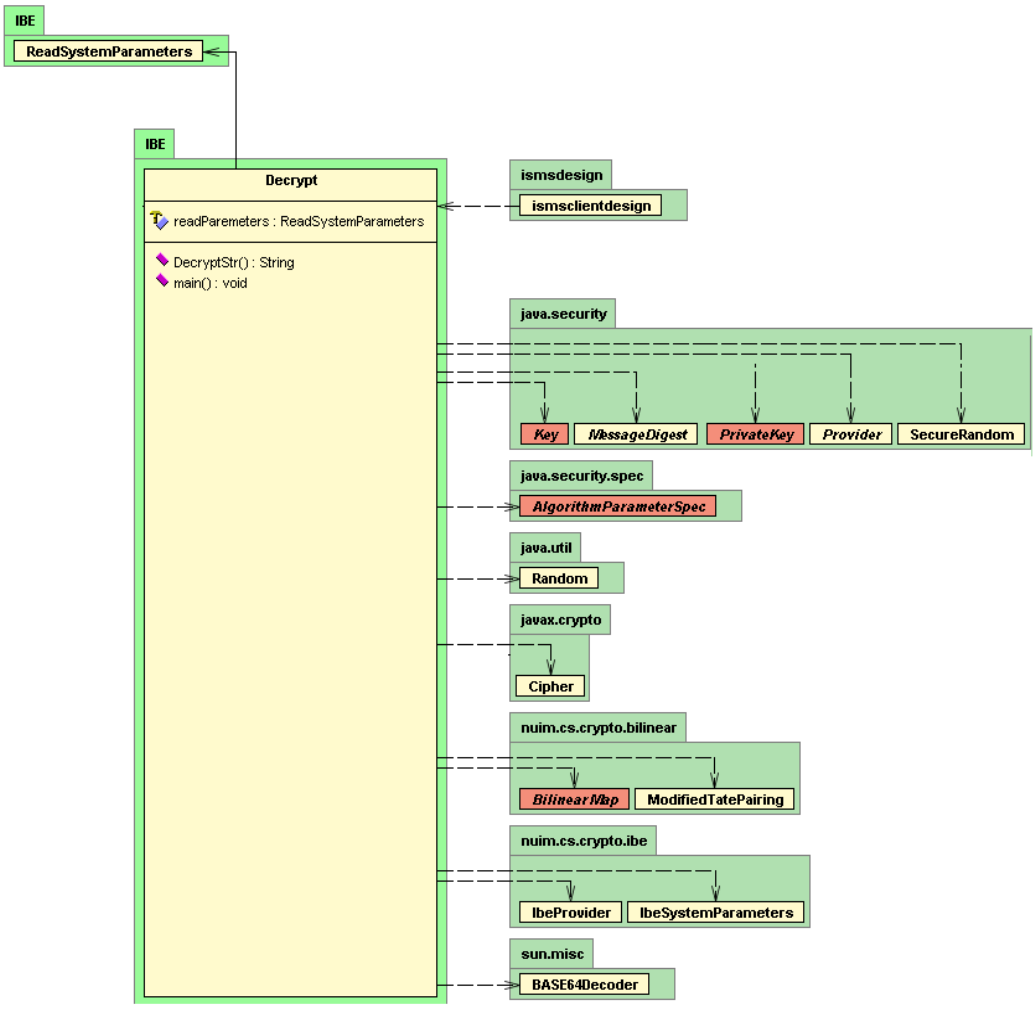


Figure 39 : IBE Decryption

Security is a like chain and it is as secure as its weakest link (Kemal Bıçakcı, IS551, September 27, 2005). Most of the time, weakest link arises from password based authentication since users of the system choose easy to remember password, write their password somewhere. By considering this issue, we apply a stronger authentication mechanism to our system of trusted third party. As it can be recognized from Figure 44, beside classical password based authentication, user needs to locate a key file which is defined when user is registered to the system (Figure 48). Without this key file, attacker could not authenticate himself to the system even he gathers username password pairs.

After user is authenticated, he can manage the firewall. Firewalls are network devices that enforce security policy of a company. They have a mechanism to allow some traffic pass in to the network while blocking other traffic. Which traffic will be allowed or not is governed by the set of rules. After detection of an attack from the IPMon logs, these rules must be arranged to block traffic coming from attacker. To manage rules of firewall, first we parse the rules with mainparser class (Figure 40).

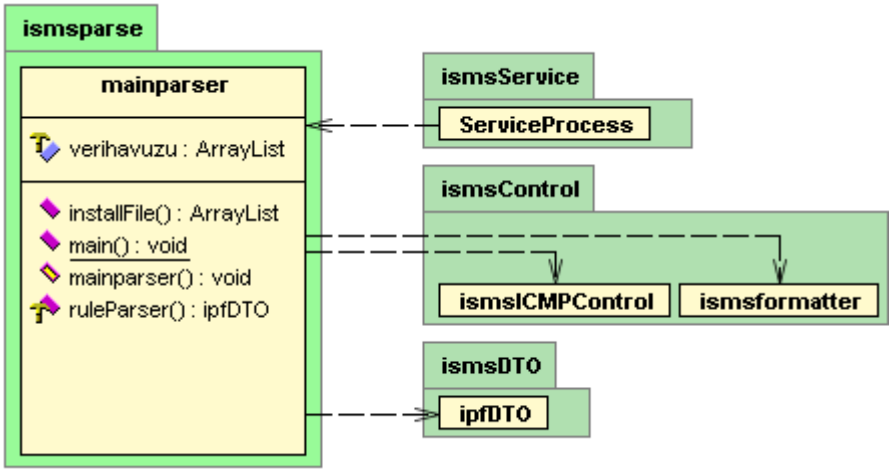


Figure 40 : Mainparser Class

Then we build an interface (Figure 46) which enables a user easily manage the firewall rules. From this interface a user can add a new rule (Figure 47), edit an existing rule (Figure 48), and change order of rules (Figure 49) and delete a rule (Figure 50).

Before the log records is started to produce at the firewall host, it needs an initial authentication key, A_0. In our implementation, the trusted third party will create random seed value and this seed value will be used on the untrusted host as an authentication key. Each time a log record is constructed at the firewall; this value is going to be used to calculate MAC of current encrypted values alone and together with previous hash. By using this MAC information, we are able to determine any deletion or alteration unassailably. Also this value is going to be

used for the single log verification which is not defined in the Schneier and Kelsey mechanism. Random seed value can be generated automatically or manually, in our implementation, we generate the seed value manually by using “Create Seed” button and send this value to the remote host. If any seed value is sent before on that day, program warns user. According to the response of warning message from user, it sends new seed value or not.

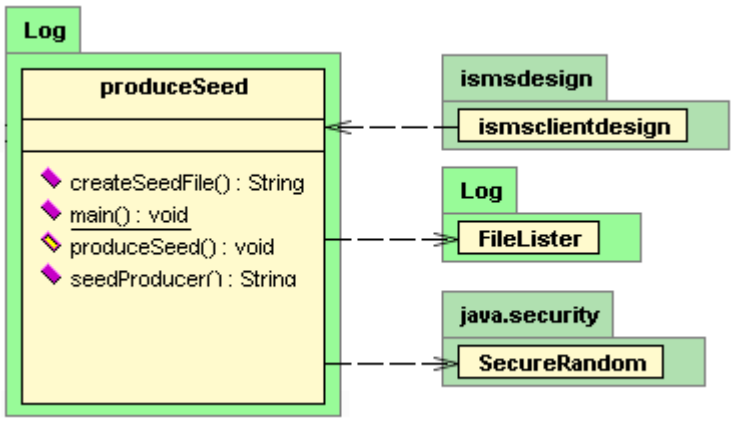


Figure 41 : Creating Seed value

Validation of logs (Figure 51) can be done in two parts:

1- Logs should be linked to each other in order to determine the missing part strictly.

2- In the case where deletion or any kind of alteration is of concern, individual log verification should be provided to make the remaining part of the logs useful.

Therefore, we first identify, whether any deletion happens on the logs or not. If any modification is of concern in one of the log record, we apply single log verification. User can apply log verification by choosing the log file and clicking “Check Logs” button.

If the administrator of trusted third party or an investigator wants to analyze log files, he clicks the “Search” button and fills required fields which are going to be searched on Figure 52 and clicks the “Submit” button. Capabilities are prepared according to authenticated user access permission and submitted fields. Then, by using these capabilities, search operation is conducted on the fields of log records. To conduct search operations, we construct searchLogFile class (Figure 42).

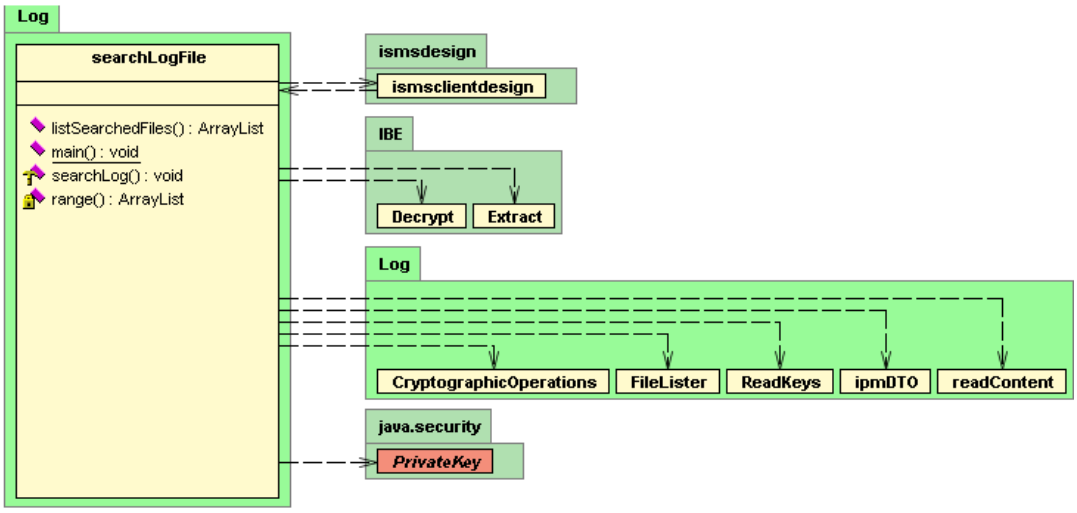


Figure 42 : Search Log File class

CHAPTER VI

CONCLUSION and FUTURE WORK

In many real-world applications, sensitive information was kept in log files on local machine and security of logs stored in plain text is provided by the underlying operating system. Because of sensitivity of information in the logs, it can provide useful information not only to the administrator of the system but also to the attacker. That is why aim of the attacker may sometimes be, rather than compromising the system it self, accessing the logs to steal confidential information or to insert a fake transaction into the logs for financial or personal gain. When the security of operating system fails, logs are under the control of attacker. Therefore, security of logs must be provided by other means. Encryption holds the promise to provide the security audit logs do require. However, it is not straightforward to satisfy other crucial requirements when logs are encrypted. Foremost of these requirements is the issue of retrieving the necessary information from the encrypted logs. In this dissertation, we have dealt with the problem of securing log files while preserving their usability. In order to overcome the problem, we have designed secure analyzable logging system and implemented it using Identity Based Encryption on IPFilter Firewall logs.

Our system comprises trusted third party, untrusted host in which logging operation realized and semi-trusted log storage server. A centralized semi-trusted log storage server provides a single location for log records and consolidates the work of examining the information.

In our system, to provide guarantees on a log's authenticity, we have applied forward integrity mechanism by augmenting it. We have considered the verification of logs in two parts: individually and as a whole. This provides robust logging system since while we are capable of detecting any alterations and deletions on the logs also we can control authenticity of the logs following the deleted record which is not considered before. As our logging system has stronger integrity preserving capability, it also provides confidentiality audit logs do require. We have encrypted log records in a way that only authorized persons can access the content of a log record.

Logs are in different types such as accounting, network traffic, logins and logouts, and dozens more. If these logs coming from diverse sources could not be analyzed effectively, having a robust logging mechanism does not have a meaning. We have constructed our encrypted log scheme by considering that logs have a specific format. We have proposed a model, while searching, this model enables us define a multiple criteria over encrypted logs and implement this model using Identity Based Encryption (IBE) and other encryption techniques on the IPFilter Firewall logs. In our mechanism, log entries and keywords extracted from these log entries are encrypted in a way that allow the investigator to determine which log entries contain a certain keyword or keyword set after receiving from the user a piece of information called a *capability* for each keyword which provide the secure delegation of search capabilities. By this way, an investigator could analyze the logs effectively but could not able to get information not related with his case. Logs which may come from different sources in different format can be analyzed and correlated to determine if an anomaly exist in the system or not by using our logging mechanism.

To recap, in this dissertation our first contribution is that adding single verification to improve Schneier and Kelsey's idea of forward integrity mechanism. As a second contribution, since logs mostly have predetermined structure, by proposing column base approach to encrypted logs, while searching, this approach enables us define multiple criteria that helps analyzing logs. Our third contribution is that in one side, our model provides secure delegation of search capabilities to authorized users while protecting information privacy, on the other, these search capabilities set boundaries of a user's search operation.

As we stated at the beginning, logs are the most fundamental resources for any security concerning case. Therefore, sharing logs for the purpose of security research is beneficial and desirable. However, because of the risk of exposing private information, strong and efficient anonymization techniques are needed for the sharing of logs. Hence, some mechanisms are proposed to share logs to prevent expose of private data [54, 55] but there is no work on the anonymization of encrypted logs. As a future work, our proposed solution on encrypted logs can be improved to help anonymization of encrypted logs.

REFERENCES

[1] Weiler, B. & Nett E. (1994). SpeedLog: A Generic Log Service Supporting Efficient Node-Crash Recovery. lEEE Micro, 60 -71

[2] Uzunay, Y. (2006). Design and Implementation of an Unauthorized Internet Access Blocking System Validating the Source Information in Internet Access Logs. Unpublished master dissertation, Middle East Technical University, Turkey.

[3] Uzunay, Y., Incebacak, D. & Bicakci, K. (2006). Towards Trustable Digital Evidence with PKIDEV: PKI based Digital Evidence Verification Model. Proceedings of the 2nd European Conference on Computer Network Defence (EC2ND)", in conjuction with the First Workshop on Digital Forensics and Incident Analysis, Springer London, United Kingdom

[4] Lessons Learned: Top Reasons for PCI Audit Failure and How to Avoid Them, VeriSign Global Security Consulting Services. Retrived November 3, 2006 from http://whitepapers.techrepublic.com.com/whitepaper.aspx?docid=243184

[5] CERT Coordination Center, CERT/CC statistics 1988-2004. Retrieved December 15, 2006 from http://www.cert.org/stats/cert_stats.html
[6] Buyer's guide for intrusion prevention systems (IPS). Retrieved November 3, 2006 from http://www.juniper.net/solutions/literature/buryer_guide/710005.pdf

[7] Jin, H. & Lotspiech, J. (2003). Proactive Software Tampering Detection. , Springer-Verlag Berlin Heidelberg, LNCS 2851, 352–365

[8] Lantz, B., Hall, R. & Couraud, J. (2006). Locking Down Log Files: Enhancing Network Security By Protecting Log Files. Issues in Information Systems, 7 (2), 43

[9] Skoudis, E. (2001). Defending Your Log Files. Retrieved November 2, 2006 from http://www.phptr.com/articles/article.asp?p=23464&seqNum=1

[10] Bishop, M. (2003). What Is Computer Security? IEEE, Security & Privacy

[11] Bishop, M. (2002). Computer Security: Art and Science, Addison Wesley Professional.

[12] Retrieved January 3, 2006 from http://www.2600.com/hacked_pages/2000/02/www.rsa.com/logo_top.gif

[13] Koblitz, N. I., (1994). A Course in Number Theory and Cryptography, 2nd edition. New York : Springer-Verlag.

[14] RSA Security's Official Guide to Cryptography. (2001). McGraw-Hill Osborne Media.

[15] Thorsteinson, P. & Ganesh G. G. A. (2003), NET Security and Cryptography Prentice Hall PTR

[16] Mao, W. (2003). Modern Cryptography: Theory and Practice, Prentice Hall PTR

[17] Mitchell, C. J. (2004). Security for Mobility. The Institution of Electrical Engineers, London, United Kingdom

[18] Boneh, D. & Franklin, M. (2001) Identity-based encryption from the Weil pairing. Advances in Cryptology - Proceedings of CRYPTO, (pp 213-229)

[19] Desmedt, Y & Quisquater, J. (1987). Public-key systems based on the difficulty of tampering. In A.M. Odlyzko(Ed), Advances in Cryptology - Proceedings of CRYPTO'86, (pp 111-117). Springer-Verlag LNCS 263

[20] Maurer, U.M. & Yacobi, Y. (1996). A non-interactive public-key distribution system. Designs, Codes, and Cryptography, 9(3), 305-316

[21] Okamoto, E. (1988). Key distribution systems based on identification information. In C. Pomerance (Ed), Advances in Cryptology - Proceedings of CRYPTO'87, (pp 194 – 202). Springer-Verlag LNCS 293

[22] Tanaka, H. (1988). In C. Pomerance (Ed), Advances in Cryptology - Proceedings of CRYPTO'87, (pp 340– 349). Springer-Verlag LNCS 293

[23] Tsuji, S. & Itoh, T. (1989). An ID-based cryptosystem based on the discrete logarithm problem. IEEE Journal on Selected Areas in Communications, 7(4): 467 – 473.

[24] Vanstone, S.A & Zuccherato, R.J. (1997). Elliptic curve cryptosystems using curves of smooth order over the ring zn. IEEE Transactions on Information Theory, 43(4):1231-1237

[25] Voltage Security: Identity-based Encryption (IT Conversation), Retrived January 18, 2006 From http://www.itconversations.com/detail.php?id=28

[26] Sakai, R., Ohgishi, K. & Kasahara, M. (2000). Cryptosystems based on pairing. In Proceedings of the 2000 Symposium on Cryptography and Information Security (SCIS 2000)

[27] Voltage Security platform (2006). Voltage Security, Inc

[28] Est´ebanez, C., C´esar, J. & Ribagorda, A. (2006). Evolving Hash Functions by Means of Genetic Programming. Proceedings of the 8th annual conference on Genetic and evolutionary computation.

[29] Bicakci, K. (2003, Sept.). On The Efficiency of Authentication Protocols, Digital Signatures and Their Applications in E-Health: A Top-Down Approach. Doctoral dissertation, Informatics Institute, Middle East Technical University, Ankara.

[30] Bellare, M. & Yee, B. S. (1997). Forward integrity for secure audit logs, tech. rep., UC at San Diego, Dept. of Computer Science and Engineering, http://citeseer.nj.nec.com/ bellare97forward.pdf.

[31] Schneier, B. and Kelsey, J. (1998). Cryptographic support for secure logs on untrusted machines, In Proceedings of the 7th USENIX Security Symposium, pages 53–62.

[32] Schneier, B. and Kelsey, J. (1999). Minimizing bandwidth for remote access to cryptographically protected audit logs. In Web Proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection.

[33] Schneier, B. and Kelsey, J. (1999). Secure audit logs to support computer forensics. ACM Transactions on Information and System Security (TISSEC), 2(2):159–176.

[34] Chong, C.N. and Peng, Z. and Hartel, P.H. (2002) Secure Audit Logging with Tamper-Resistant Hardware. Technical Report TR-CTIT-02-29 Centre for Telematics and Information Technology, University of Twente, Enschede.

[35] Accorsi, R. & Hohl, A. (2006). Delegating secure logging in pervasive computing systems, 3rd Conf. Security in Pervasive Computing.

[36] Waters, B. R., Balfanz, D., Durfee, G., & Smetters, D. K. (2004). Building an Encrypted and Searchable Audit Log. In Proceedings of the Network and Distributed System Security Symposium, NDSS 2004, San Diego, California, USA.

[37] Davis, D., Monrose, F. & Reiter, M. K. (2004). Time-Scoped Searching of Encrypted Audit Logs. In Javier Lopez, Sihan Qing, and Eiji Okamoto (Ed). Information and Communications Security, 6th International Conference, ICICS 2004, Malaga, Spain.

[38] Ohtaki, Y., (2005). Constructing a searchable encrypted log using encrypted inverted indexes. International Conference on Cyberworlds (CW'05)

[39] Song, D. X., Wagner, D., & Perrig, A. (2000). Practical Techniques for Searches on Encrypted Data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P 2000), 44–55

[40] Goh, E. (2003). How to search efficiently on encrypted compressed data. In the Cryptology ePrint Archive, Report 2003/216. http://eprint.iacr.org/2003/216/

[41] Chang, Y. C. & Mitzenmacher, M. (2005). Privacy Preserving Keyword Searches on Remote Encrypted Data. In John Ioannidis, Angelos D. Keromytis, & Moti Yung (Ed). Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA.

[42] Goh, E-J. (2004). Secure Indexes. Cryptology ePrint Archive, Report 2004/016. Available at http://eprint.iacr.org.

[43] Boneh, D., Crescenzo, G. D., Ostrovsky, R. & Persiano, G. (2004). PublicKey Encryption with Keyword Search. In Christian Cachin and Jan Camenisch (Ed). Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland.

[44] Golle, P., Staddon, J. & Waters, B. R. (2004). Secure Conjunctive Keyword Search over Encrypted Data. In Markus Jakobsson, Moti Yung, and Jianying Zhou (Ed), Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, Chin.

[45] Park, D. J., Kim, K. & Lee, P. J. (2004). Public Key Encryption with Conjunctive Field Keyword Search. In Chae Hoon Lim and Moti Yung (Ed). Information Security Applications: 5th International Workshop, WISA 2004, Jeju Island, Korea.

[46] Krutz, L. R. & Vines, D. R. (2003). The CISM Prep Guide: Mastering the Five Domains of Information Security Management, John Wiley & Sons

[47] Merrill, C. R. (2000). Time is of the Essence, CIO.com

[48] Ayrapetov, D., Ganapathi, A. & Leung, L. (2004). Improving the Protection of Logging Systems. UC Berkeley Computer Science

[49] Kevin, L., David, L. & Ben, S. (2004). Assessing Network Security. Microsoft Press

[50] Keeney, M. (2005). Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors. Doctoral dissertation, http://www.secretservice.gov/ntac_its.shtml.

[51] Owens, L., Duffy, A. & Dowling, T. (2004). An Identity Based Encryption System. http://www.crypto.cs.nuim.ie

[52] http://www.bouncycastle.org/

[53] Martin, R. C. (2003). UML for Java™ Programmers, Prentice Hall PTR.

[54] Slagell, A. J. Wang, W. Y., (2004). Network Log Anonymization: Application of Crypto-PAn to Cisco NetFlows", NSF/SFRL Workshop on Secure Knowledge Management (SKM)

[55] Slagell, A., Li, Y. & Luo, K. (2005). Sharing Network Logs for Computer Forensics: A New tool for the Anonymization of NetFlow Records, Computer Network Forensics Research Workshop, held in conjunction with IEEE SecureComm.

APPENDICES

APPENDIX A. SCREEN SNAPSHOTS of I-SMS

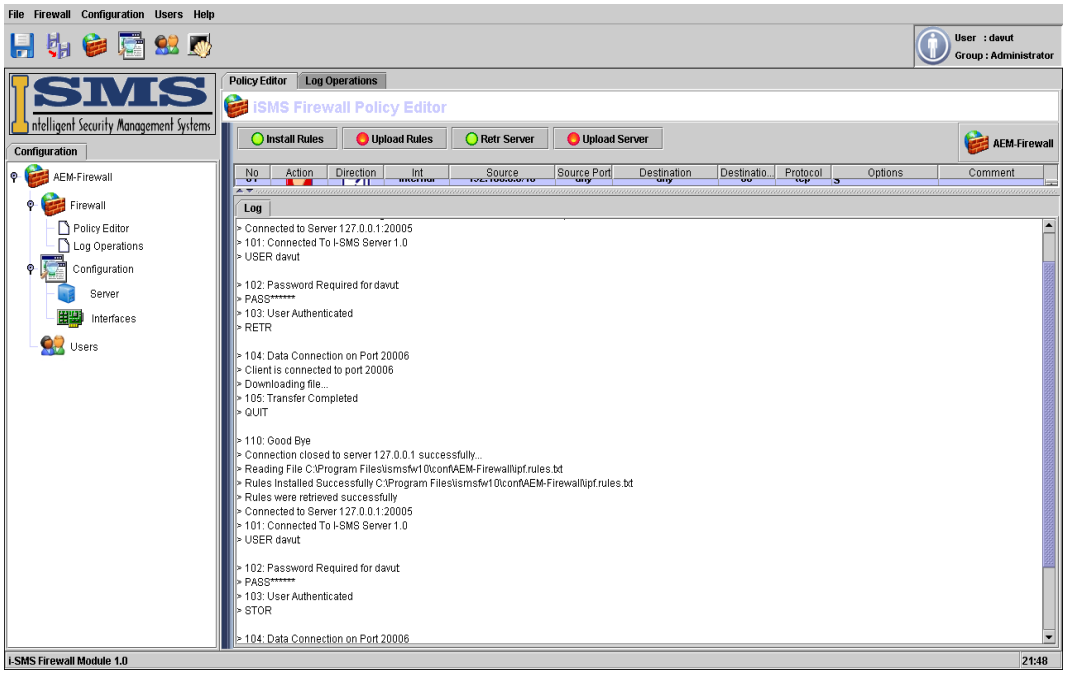


Figure 43 : An Example of Protocol Process

Figure 44 : Login to System

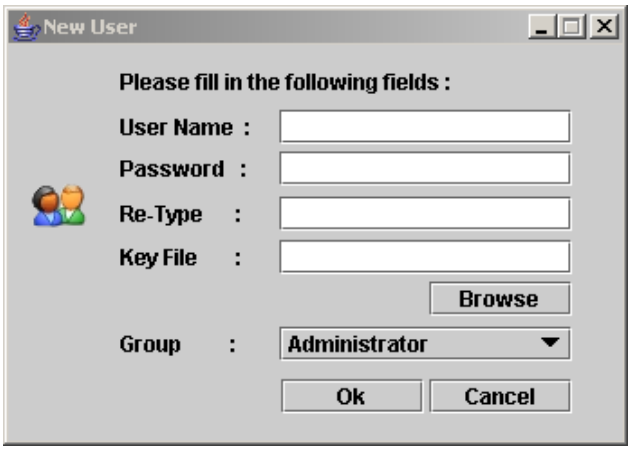


Figure 45 : Add new User

Figure 46 : Policy Editor

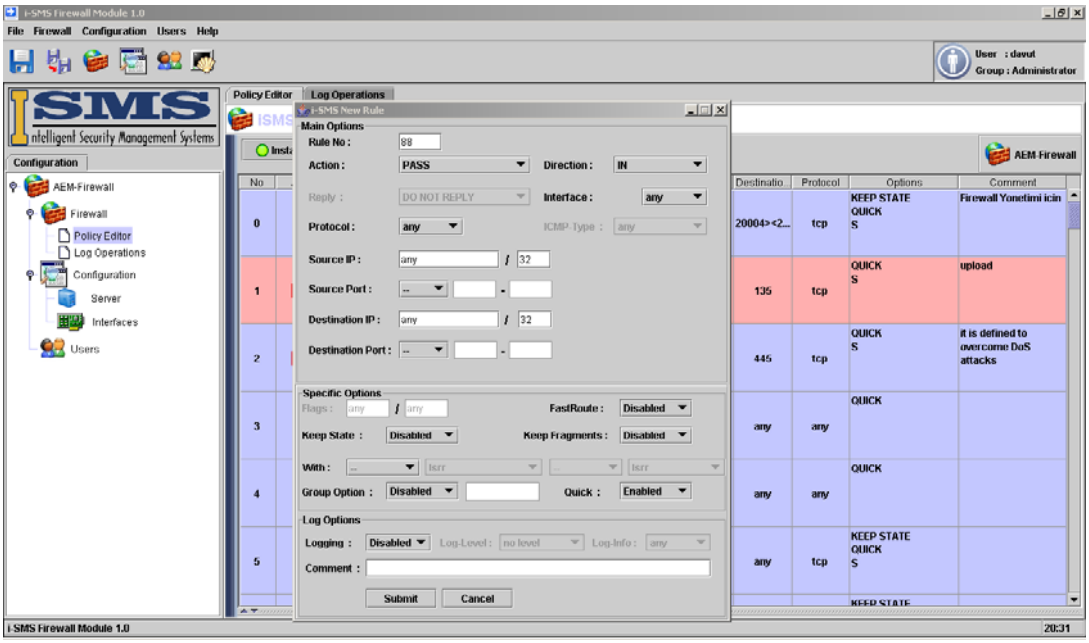


Figure 47 : Add new rule

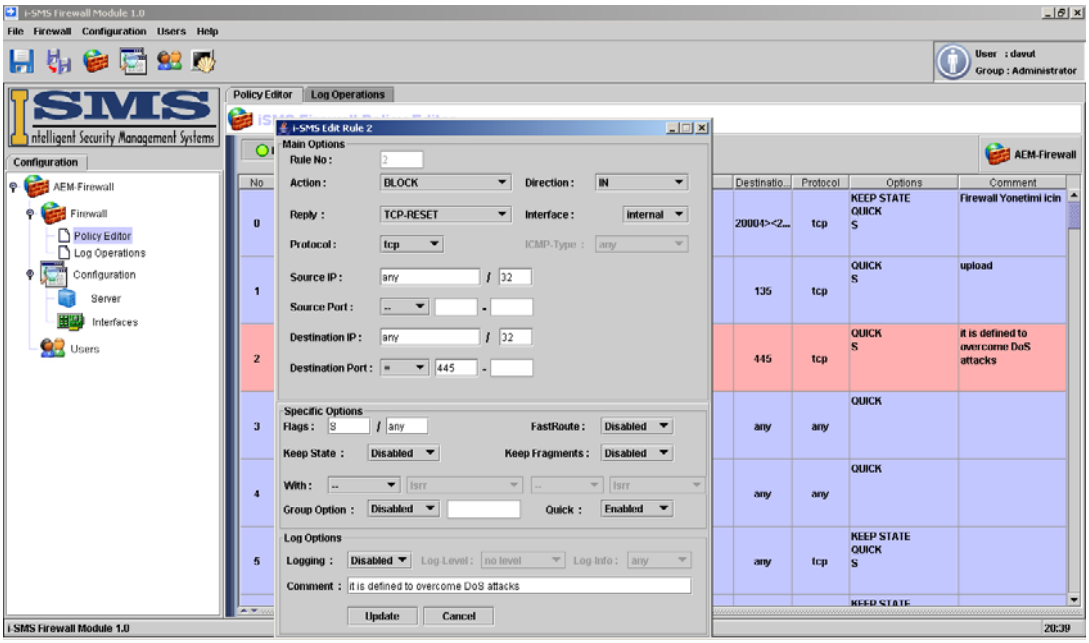


Figure 48 : Edit rule

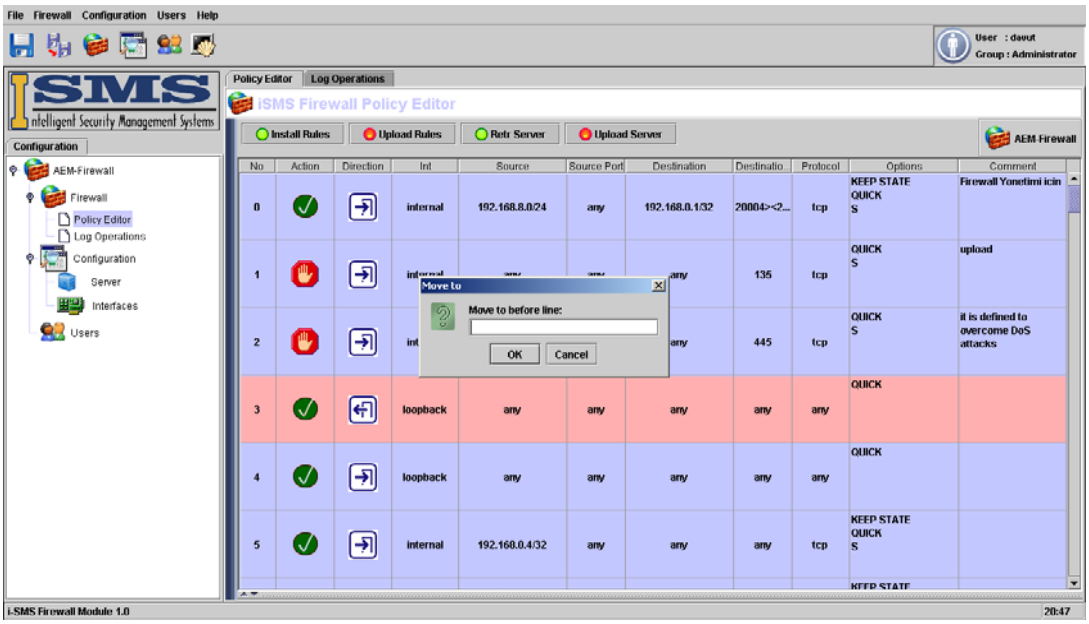


Figure 49 : Change order of rules

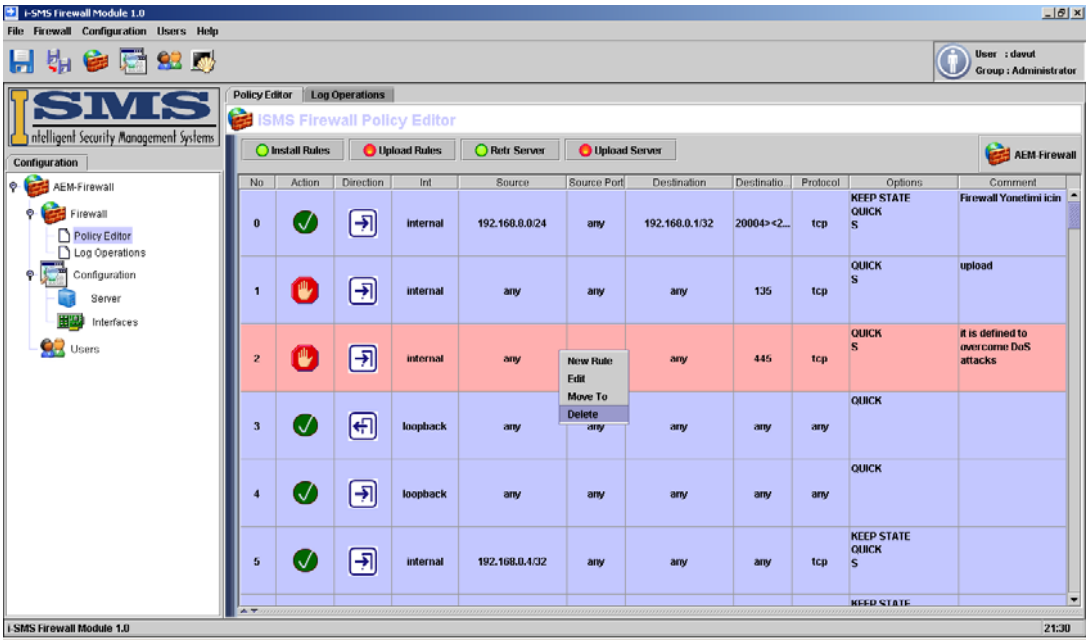


Figure 50 : Delete a rule

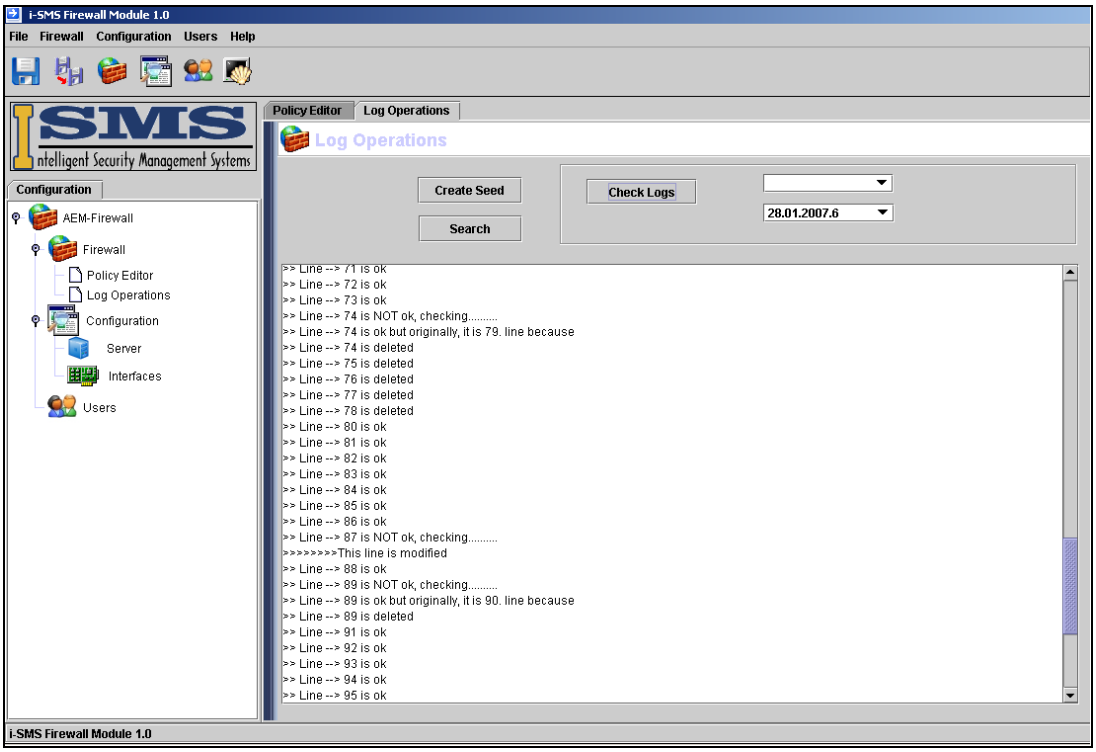


Figure 51 : Log validation

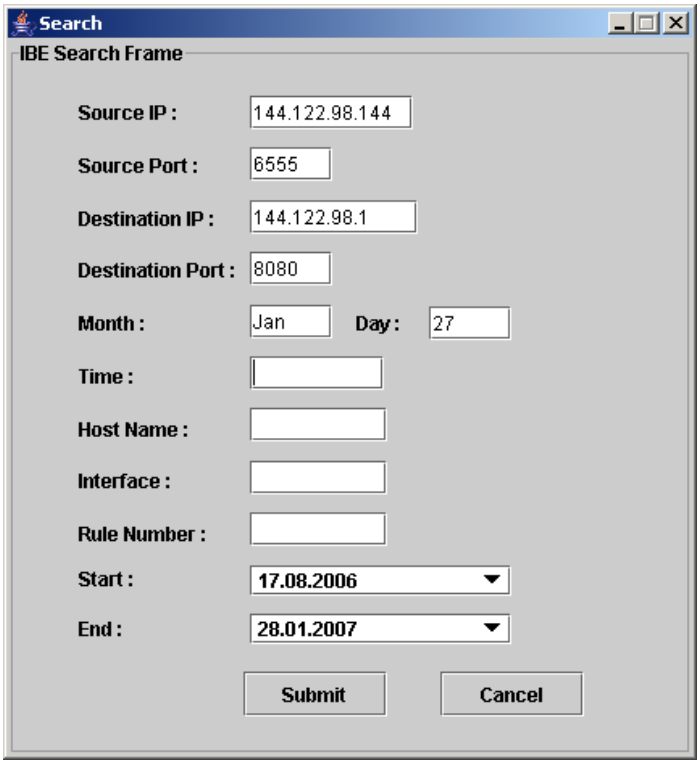


Figure 52 : IBE Search Frame

APPENDIX B. IP FILTER (IPF) FIREWALL RULES

IPF consists of one file which includes the rules. The file is read from top to bottom, and if a packet matches a rule, the firewall does NOT stop parsing the file. This is called “The last matching rule always takes precedence.” Rule processing can be controlled by using the “quick” word. If a rule includes “quick” and a packet match with that rule, it stops the parsing the file.

1. Basic Rules:

“block in all ” blocks the all packets
“pass in all ” passes the all packets

2. Filtering by IP address:

“block in quick from 120.45.0.0/16 to any ” this rule will block the packets coming from the 120.45.0.0/16 network and pass other packets.

3. Interface Control:

Every packets come from one interface and goes to another interface.
“block in quick on xl0 all” this rule blocks the packets the coming from xl0 to any other interfaces.
“pass in quick on lo0 all ” this rule pass the packets coming from lo0 interface to any other interfaces.
“block in quick on x10 from 144.122.0.0/16 to any” this rule says that it’s only blocked if it comes in on the x10 interface from the 144.122.0.0/16 network

4. The "out" Keyword:

“in” keyword indicates the packets coming outside to our network, but by using the “out” keyword, a packet can be controlled, while it is going out from our network.
“pass out quick on xl0 from 144.122.98.0/16 to any” this rule says that if a packet going out from the xl0 interface in the 144.122.98.0/16 network, let it go out.

5. The "log" Keyword:

When we want to see what happened, according to a rule, we use “log” keyword.
“block in log quick on lo0 from 144.122.98.0/24 to any” this rule blocks the packets coming from the lo0 in the 144.122.98.0 network and log the actions, matching events to this rule.

6. The "proto" Keyword:

By using the proto keyword, which protocol will be taken care is defined.
“block in log quick on lo0 proto icmp from any to any”

7. The "icmp-type" Keyword:

There are many of ICMP type and by using proto keyword, blocking all ICMP packets can be not good. By using “icmp-type” keyword, it can be decided that the rule is applied to which “icmp-type” traffic. For example, for the “ping” and “trace route”, it should be let in ICMP types 0 (ECHO_REPLY) and 11 (time exceeded).

“pass in quick on lo0 proto icmp from any to 144.122.98.0/24 icmp-type 0 ” an ICMP type 0 packet from 144.122.98.0/24 will get passed by this rule.

8. The "port" Keyword:

Most services works on specific ports. By specifying the rule for that ports, we can decide, let that service or not.

“block in log quick on lo0 proto tcp/udp from any to 144.122.98.0/24 port = 111 ” any tcp or udp packets goes to 144.122.98.0/24 port=111 will be blocked and logged.

9. The "keep state" Keyword:

Keeping state allows ignoring the middle and end and simply focusing on blocking/passing new sessions. If the new session is passed, all its subsequent packets will be allowed through. If it’s blocked, none of its subsequent packets will be allowed through. For keeping states of the session a state table is used. An example for running an ssh server (and nothing but an ssh server):

If our rule set like this:

block out quick on lo0 all
pass in quick on lo0 proto tcp from any to 144.122.98.1/32 port = 22 keep state

If we don’t use the “keep state” keyword, we could not make a ssh connection because of first rule. First rule blocks all packets to go out from network but by using “keep state” keyword, we say implicitly, allow packets, which comes to the 144.122.98.1 computer for ssh connection, to go out.

10. The "flags" Keyword:

pass in quick on lo0 proto tcp from any to 144.122.98.1/32 port = 23 keep state
pass out quick on lo0 proto tcp from any to any keep state
block in quick all

block out quick all

The problem in the above rule set is that it's not just SYN packets that're allowed to go to port 23, any old packet can get through. We can change this by using the flags option:

pass in quick on tun0 proto tcp from any to 144.122.98.1/32 port = 23 flags S keep state
pass out quick on tun0 proto tcp from any to any flags S keep state
block in quick all
block out quick all

Now only TCP packets, destined for 144.122.98.1/32, at port 23, with alone SYN (flags S represents the SYN) flag will be allowed in and entered into the state table. Alone SYN flag is only present as the very first packet in a TCP session (called the TCP handshake).

11. The "keep frags" Keyword:

With the "keep frags " keyword, IPF will notice and keep track of packets that are fragmented, allowing the expected fragments to go through.

"pass in quick on rl0 proto tcp from any to any port = 80 flags S keep state keep frags "

12. Responding To a Blocked Packet:

When a service isn't running on a Unix system, it normally lets the remote host know with some sort of return packet. In TCP, this is done with an RST (Reset) packet. Responding a packet give the information to the attacker, so, we can mislead the attacker by sending not real return packet.

block return-rst in log quick on rl0 proto tcp from any to any
block return-icmp-as-dest (port-unr) in log quick on rl0 proto udp from any to any

Block and log all remaining traffic coming into the firewall
- Block TCP with a RST (to make it appear as if the service isn't listening)
- Block UDP with an ICMP Port Unreachable (to make it appear as if the service isn't listening)

13. Rule Groups:

By assuming that we have two interfaces in our firewall with interfaces xl1, and xl2, Rule Groups can be explained.

xl1 is connected to our "DMZ" network 144.12.9.64/26
xl2 is connected to our protected network 144.12.9.128/25

block out quick on xl1 all head 10
pass out quick proto tcp from any to 20.20.20.64/26 port = 80 flags S keep state group 10
block out on xl2 all
In this example, If the packet is not destined for xl1, the head of rule group 10 will not match, and we will go on with our tests. If the packet does match for xl1,the quick keyword will short-circuit all further processing at the root level, and focus the testing on rules which belong to group 10; namely, the SYN check for 80/tcp.

14. The "Fastroute" Keyword:

Firewall forwards some packets, and blocks some other packets, so it is like a well behaved router which decrements the TTL on the packet. This presents that there is firewall there. But presence of firewall can be hidden from some applications like unix trace route which uses UDP packets with various TTL values to map the hops between two sites. If we want incoming trace routes to work, but we do not

want to announce the presence of our firewall as a hop, we can do it by using “fastroute” keyword.

block in quick on xl0 fastroute proto udp from any to any port 33434 >< 33465