PRIVACY-PRESERVING ALERT CORRELATION AND REPORT RETRIEVAL

BEN WEN ZHU

A THESIS

IN

THE CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN INFORMATION SYSTEMS SECURITY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

JANUARY 2010

Your file Votre référence
ISBN: 978-0-494-67295-2
Our file Notre référence
ISBN: 978-0-494-67295-2

ABSTRACT

Privacy-Preserving Alert Correlation and Report Retrieval

Ben Wen Zhu

Intrusion Detection Systems (IDSs) have been widely deployed on both hosts and networks and serve as a second line of defense. Generally, an IDS flags malicious activates as IDS alerts and forwards them to security officers for further responses. The core issue of IDSs is to minimize both false positives and false negatives. Previous research shows that alert correlation is an effective solution. Moreover, alert correlation (in particular, under the cross-domain setting) can fuse distributed information together and thus be able to detect large-scale attacks that local analysis fails to handle. However, in practice the wide usage of alert correlation is hindered by the privacy concern.

In this thesis, we propose the TEIRESIAS protocol, which can ensure the privacy-preserving property during the whole process of sharing and correlating alerts, when incorporated with anonymous communication systems. Furthermore, we also take the fairness issue into consideration when designing the procedure of retrieving the results of correlation. More specifically, a contributor can privately retrieve correlated reports in which she involved.

The TEIRESIAS protocol is based mainly on searchable encryption, including both symmetric-key encryption with keyword search (SEKS) and public-key encryption with keyword search (PEKS). While designing TEIRESIAS, we identify a new statistical guessing attack against PEKS. To address this problem, we propose the PEKSrand scheme, which is an extension of PEKS and can mitigate both brute-force guessing attacks and statistical guessing attacks. The PEKSrand scheme can either be used independently or be combined with TEIRESIAS to further improve its privacy protection.

Acknowledgments

My sincere thanks go to people who provide help to finalize this thesis.

First of all, I would like to express my sincere appreciation to my advisor, Dr. Bo Zhu, for giving this opportunity to work under his supervision. I am very grateful for his academic guidance, invaluable suggestions and constant support, without which this thesis could not be possible.

Meanwhile, I am very grateful to the professors in the thesis examining committee, Dr. Anjali Awasthi, Dr. Lingyu Wang and Dr. Dongyu Qiu for proving precious comments to this thesis.

Lastly, I would like to express my eternal gratitude to my family for their constant encouragement and endless support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The major goal of this thesis is to design a protocol that can provide secure and privacy-preserving (intrusion detection) alert correlation under the cross-domain settings.

To defend against various attacks, Intrusion Detection Systems (IDSs) deployed on hosts and networks as a second line of defense. IDSs can output alerts when suspicious activities are observed. To take full advantage of IDS alerts, we may correlate them so as to help us better understand the security threats and take appropriate responses. Such a method is effective within a single trust domain, e.g., a large enterprise-wide network. However, previous solutions adopted in a single domain are incompetent in detecting cross-domain distributed attacks and more delicate cross-domain multi-hop attacks (e.g., stepping stone attacks [37, 77, 81]). Therefore, to further enhance the ability of detecting malicious behaviors, it is desirable to achieve cross-domain alert correlation.

In reality, unfortunately, there is slow progress in sharing IDS alerts across domains. The major barrier is the concern that, shared alerts may contain sensitive information, which may be disclosed and misused at a later time. In this thesis, we leverage recent advances in searchable encryption [15, 68] and keyed-hash functions to address the privacy and utility issues in the cross-domain alert correlation. Moreover, in the course of developing our solution, we identify a new type of attacks against a well-known searchable encryption scheme, i.e., Pubic-key Encryption with Keyword Search (PEKS) [15]. An extension of PEKS is then proposed to protect PEKS against both the new attack (called statistical guessing attacks) and other attacks indicated in previous work [3,8,18,60].

This chapter gives a brief introduction about the background and the motivation of our research, together with a summary of our contributions.

1.1 Intrusion Detection and Alert Correlation

In traditional networks, firewalls act as the first line of defense. Through enforcing predetermined rules, firewalls can limit potential paths that adversaries utilized to access internal networks and block certain attacks directly. However, it is far from being able to prevent and monitor all types of intrusions. Thus, Intrusion Detection Systems (IDSs) are used as a second line of defense, i.e., detecting intrusions and triggering prompt responses that recover compromised systems and/or update systems to prevent similar attacks in the future.

Generally speaking, intrusion detection techniques can be classified into two categories: *misuse detection* and *anomaly detection*. Misuse detection is based on patterns of well known intrusions (signatures). When comparing suspected activities' profiles with signatures, if an activity matches a known pattern, an alert will be generated accordingly. The main advantage of misuse detection is the low ratio of false positives. However, it can only detect attacks with known signatures. Anomaly detection generates alarms when an IDS observes activities that abnormally deviate from the recognized normal profiles which are established during the training process. The main advantage of anomaly detection is its ability to detect unknown attacks. However, a major weakness of anomaly detection is its relatively high false positive rate.

Although extensive research has been conducted to build more effective IDSs, current

IDSs still suffer from many open problems. For example, an IDS usually produces a large number of alerts. The high volumes and low quality of the raw alerts make it very difficult for security administrators to identify those important alerts and respond to them in a timely manner. Another open challenge is to detect large-scale distributed attacks and cross-domain multiple-step attacks. To address these two problems, alert correlation techniques are developed.

Instead of measuring the validity of each single alert, alert correlation techniques focus on discovering the logical relationships among alerts outputted by IDSs. In recent years, several alert correlation techniques have been proposed to facilitate the analysis of intrusion alerts. These techniques can be roughly divided into two categories: *template-based* alert correlation, which can only correlate alerts satisfying pre-defined attack scenario templates, and *condition-oriented* alert correlation which is not only able to correlate alerts satisfying a pre-defined template, but also can construct new attack scenarios.

In this thesis, we are interested in more powerful condition-oriented alert correlation. A typical work in this category is the alert correlation scheme proposed by Ning et al. [51] (a more detailed overview of [51] is to be presented in Section 2.3). The idea of their approach is that, each alert is treated separately, and both the needed conditions (A.K.A. prerequisites or pre-conditions) to trigger an attack and the possible conditions (A.K.A. consequences or post-conditions) caused by this attack are considered. For example, an attacker has to install a Distributed Denial of Service (DDoS) daemon program in vulnerable hosts before launching a DDoS attack. In this case, exploiting hosts and installing DDoS daemons can be viewed as prerequisites of a DDoS attack, and the DDoS attack is the consequence of either exploiting a host or installing a DDoS daemon on the host.

1.2 Privacy-preserving Alert Sharing and Motivation of Our Work

A report released by Symantec [69] shows that there exists a trend of increasing professionalization and commercialization of malicious activities, as well as the increasing number of attacks. Malicious users have built underground forums (e.g., IRC or web sites) for exchanging information about new exploits and underground economy servers to trade compromised resources. While malicious parties are collaborating with each other so as to launch attacks more efficiently and effectively, unfortunately, on the other side of the attacking-defending competition, there are relatively less collaborations going on among the defenders.

Recently, more and more researchers, companies, organizations, and even governments have begun to realize the tremendous benefits that may be brought to the detection of malicious activities, through establishing such collaborations. In particular, sharing security data among various detection systems, e.g., cross-domain Intrusion Detection Systems (CIDS), is one of the most promising types of collaborations. It has significant impacts on many security applications, e.g., detecting zero-day attacks and multi-hop attacks, or generating a large-scale picture of Internet security status and trends. In reality, however, there is slow progress in sharing security data across domains. The major barrier is the concern that, shared security data may contain sensitive information, which may be disclosed and misused at a later time. And such information may be originally difficult to obtain by adversaries or potential competitors that participate in the data sharing system. For example, based on the shared security alerts, adversaries can deduce the locations of IDS sensors through fingerprinting attacks [44].

There has been extensive work in privacy-preserving intrusion detection systems or alert sharing [6, 11, 45–48, 64, 79]. Most algorithms/schemes proposed are effective in preserving privacy while detecting high-volume events but failing to perform more delicate tasks, e.g., alert correlation [51]. To the best of our knowledge, the only work addressing this issue was done by Xu and Ning [78,79]. However, since the matching of events is based on causal relations, together with the usage of an aggregation algorithm, their scheme has a relatively high false positive, which has limited its practical usages.

Another major problem of previous solutions for privacy-preserving alert sharing is that, most of them aim at localized data sharing within a company or an organization, and thus assume the existence of either a fully-trusted online centralized entity that is responsible for collecting and analyzing security data from different sources, or a fully-trusted relationship among the group members that share their security data. Both assumptions are implausible when considering the task of cross-domain alert correlation. The major problem of maintaining a fully-trusted online centralized entity is the single-point-of-failure in terms of both availability and security. If the online centralized entity is compromised, with the secret acquired, the adversary can trivially terminate the service and/or break the privacy protection [63]. The availability issue can be addressed by providing a back-up scheme. Unfortunately, it does not help in security. In all previous centralized solutions, at any time at least one fully-trusted centralized entity has to be online, because the task of analyzing security data relies on the secret possessed only by the trusted entities. As to the latter assumption, i.e., the fully-trusted relationship among members, it does not hold when the insider attack is possible, e.g., in the scenarios that participants of the system are potential competitors.

Previous work in privacy-preserving alert sharing mainly concentrates on the core task of correlation. However, we argue that the privacy-preserving property should be held during the whole process of collaboration, which includes alert collection, alert correlation, and the retrieval of correlation results. Currently, there are two main methods for the acquisition of correlation results. Either the centralized entity is responsible for both correlating alerts and returning the results to all legitimate users or a specific set of users (e.g., users who may be affected by the detected attack), or it is only responsible for alert correlation and publishes the results upon the completion of correlation online. Apparently, it is inefficient to forward the results of correlation to every member, despite of their relevance. In terms of privacy, in the first method, if the results are only forwarded to those members who are relevant, the identity of the contributors has to be disclosed to the centralized entity. As to the later, due to the results' open access to everyone, it is vulnerable to probe-response attacks [10,46,62].

Another issue largely ignored in previous work in privacy-preserving alert sharing is fairness. Many previous solutions work well under an implicit assumption that, participants are motivated to share their data in exchange of more accurate analysis results and thus more effective defense. It makes sense when there is no selfish participant. Since there is a fundamental trade-off between privacy and usability [63], whenever any usable information is disclosed, it must have weakened the privacy protection to a certain extent. Thus, a selfish participant may be reluctant to provide any useful data. If there is no mechanism that can ensure the fairness in terms of the contributions and rewards, the system is vulnerable to attacks similar to Freeriding in peer-to-peer systems [4]. Under cross-domain settings, neither there exists a fully-trusted online centralized entity, nor it is possible to enforce different entities to unconditionally share their security data, even if it is the government (or a designated department of the government) that enforces the collaboration for the sake of national security [52]. Without sufficient incentives together with a mechanism enforcing fairness and privacy at the same time, the quantity and quality/usability of the shared data are questionable. To maintain a fair correlation system, an intuitive solution is that, the more useful information that a participant shares with others, the more useful analysis results should be accessible to her/him.

1.3 Research Contributions and Thesis Organizations

The main contribution of this research is the designing of a secure and privacy-preserving system that encourages the sharing of security data. The main characteristics of our design include:

- Enable privacy-preserving alert correlation which can detect both high-volume events and more delicate events (e.g., multi-hop attacks).
- Do not require the existence of a fully-trusted online centralized entity. Instead, our scheme needs one or a few online semi-trusted servers, which are honest-but-curious [33].
- Provide privacy protection during both alert collection, correlation and retrieval.
- Ensure the fairness of alert sharing. To the best of our knowledge, our work is the first one that can ensure both fairness and privacy during alert sharing and retrieval in the absence of an online fully-trusted entity.
- Be resistant to probe-response and collusion attacks launched by outsiders and insiders.
- Extend PEKS to support predicate privacy based on the idea of randomization. To the best of our knowledge, this is the first work that ensures predicate privacy in the public-key settings without requiring interactions between the receiver and potential senders, the size of which may be very large.
- Identify a new type of attacks against PEKS, i.e., statistical guessing attacks.
- Introduce a new concept of statistical privacy, and extend PEKS to support this new privacy property.

The rest parts of this thesis are organized as follows. In Chapter 2, we review the related research works and define the privacy problems touched in this thesis. In Chapter 3, we describe the framework and general ideas of our design. Afterwards, we present defined privacy-preserving alert correlation protocol named TEIRESIAS in Chapter 4, followed by the implementation and experimental results in Chapter 5. The predicate privacy-preserving extensions to the PEKS are presented in Chapter 6. The conclusion is drawn in Chapter 7.

Chapter 2

Literature Review

In this chapter, we review previous research that is relevant to our work. It mainly includes alert correlation, privacy-preserving information sharing, and predicate privacy in PEKS.

2.1 Alert Correlation

Before digging deeper into alert correlation, it is worthwhile to distinguish between alert fusion and alert correlation within the context of CIDS due to the natural differences and connections between them.

2.1.1 Alert Fusion and Alert Correlation

Alert fusion can be considered as the process of mitigating redundant alerts which are triggered by the same attacking activity or the process of grouping similar alerts (e.g., alerts with the same source and destination IP address). In contrast, alert correlation attempts to organize alerts into some kind of high-level (e.g., the trace of multi-step attacks) and easy understood models (e.g., visualized attack scenarios or attack graphs). In other words, compared to alert fusion, alert correlation focuses on identifying the relationships among collected alerts and extracting attackers' strategies.

Valdes and Shinner proposed a probabilistic approach to fuse alerts by estimating the similarities of alert attributes [72]. The similarity is evaluated based on the feature-specific similarity functions of all investigable attributes. In addition, clustering techniques, through which similarity is evaluated, have also been adopted to fuse alerts. Cuppens [20] fused similar alerts into clusters, and Julish et al. [41,42] used a hierarchical conceptual clustering technique to fuse alerts. Recently, Gu et al. [36] proposed a decision-theoretic alert fusion technique based on the likelihood ratio test (LRT). In [73], Valeur et al. put alert fusion into a comprehensive intrusion detection alert analysis framework.

Note that, alert fusion and alert correlation are complementary functions. Hence, our work can be easily integrated with any alert fusion techniques. In the remained parts of this thesis, we assume that the input to our protocol is fused alerts.

2.1.2 Review of Alert Correlation

The goal of alert correlation is neither to significantly reduce the number of alerts, nor to roughly classify alerts into different groups. Instead, it is to link relevant alerts and reconstruct attack scenarios from isolated alerts. Alert correlation serves the role of helping security professionals better understand what has happened, how it happened and even what is happening and how it is evolving on the network. Previous works on alert correlation can be divided into the following two categories:

- **Template-based Alert Correlation** Formal methods have been widely employed in most previous works of this category, such as LAMBDA [21], STATL [25], ADeLE [71], all of which are formal attack specification languages defined to facilitate alert correlation. Meanwhile, a data mining approach is used in [22] to achieve the same goal. The effectiveness of template-based correlation is limited by pre-defined templates which are specified by humans or are learned automatically from training datasets. In particular, they are able to discover *known* attack scenarios but fail to uncover new attack strategies.

- **Condition-oriented Alert Correlation** To overcome the limitation of template-based correlation, researchers proposed to correlate alerts based on the casual relationship between alerts. The most common feature of these approaches is that, alerts are correlated based on pre-condition (i.e., prerequisites, which are needed conditions to trigger this attack) and post-condition (i.e., consequences, which are all possible conditions that can be caused by this attack) of each individual alert. The well-known works in this category include correlation algorithms proposed by Templeton and Levis [70], Cuppens and Miège [30], and Ning et al. [51].

  JIGSW [70] is the first work that considers using the idea of condition-oriented alert correlation. However, it is not a very flexible choice when used in practice. For instance, it considers *full* correlation, which requires all the prerequisites of an attack happen in order to consider its consequence. In particular, this may lead to correlation missing if IDS fails to identify any of the prerequisite attacks. Other weaknesses include ignoring failed attempts and lacking of aggregating consideration [51].

  To overcome the drawback of mis-detection or mis-reporting in full correlation, the algorithms proposed in [30, 51] employ *partial* correlation, which only requires to satisfy part of the pre-conditions and post-conditions. Because an instance/application of our work is built upon the correlation techniques proposed by Ning et al. [51], we will give a more detailed review of their work in Section 2.3.

2.1.3 Cross-domain Alert Correlation

A typical framework for cross-domain alert correlation is shown in Figure 1, where *alert collection* aims to gather alerts from heterogeneous IDSs, and *alert fusion* process combines alerts triggered by the same attack from independent IDSs, while *alert correlation* discovers attack threads among alerts. Finally, *visualization* and *result delivery* are two optional processes highly beneficial in understanding analysis results. *Visualization* gives a logical and clear presentation of detailed attack scenarios and attack strategies. *Result delivery* aims to help contributed IDSs' profit from their sharing by enlarging the scope of a contributor from local network to global network.

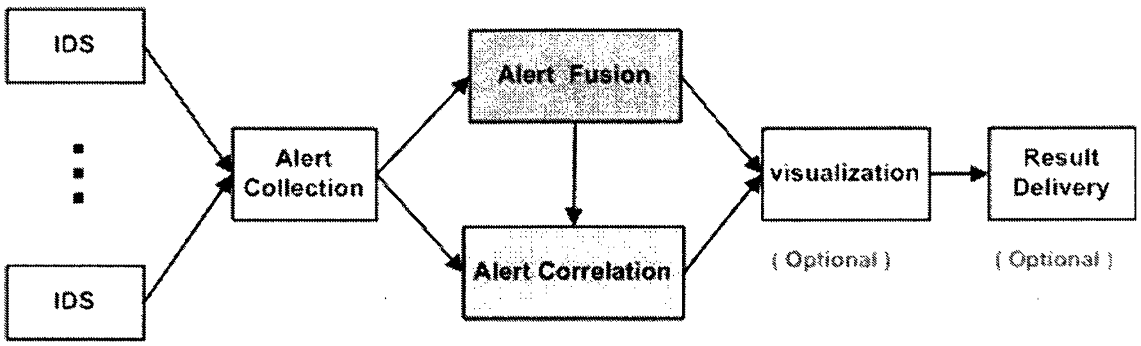


Figure 1: A Cross-domain Alert Correlation Framework

2.1.4 Privacy Concerns in Alert Correlation

There are various privacy concerns in cross-domain alert correlation, and are usually relevant to the attributes of IDS alerts. The two attributes of particular interests to our work are *identity* and *type*.

- **Identity** Most attackers interest in information related to identities, which is the straight forward privacy offense. If an attacker can extract the identity related information encoded in a alert, no matter how little other information obtained, he can conclude that certain security vulnerabilities exist on that particular machine. In addition, without identity protection, attackers can easily distinguish the alerts related to the specific interested machines by excluding irrelevant alerts. The identity information may also be utilized by skilled and active attackers to compose special fingerprints to probe the security posture of a particular machine. i.e., probe-response attacks [10,46,62]. In an IDS alert, identity information is usually reflected by Source IP address, where the network traffic (e.g. a attack) is initiated and Destination IP address, where the network traffic reaches.

- **Type** Attackers are also interested in the information revealed through an alert's type attribute, which may indicate a successful launched attack or expose type or version information of the IDS through the particular type signatures. Thus, alert types may directly reveal the existing vulnerabilities and security postures of the hosts or networks.

2.2 Privacy-Preserving Security Information Sharing and Alert Correlation

In this section, we first review previous research work about privacy-preserving security information sharing in general, and then focus on the related work in privacy-preserving alert correlation in particular.

2.2.1 Privacy-Preserving Security Information Sharing

The concerns about privacy in intrusion detection rose as early as the late 1990s [67]. In [48, 49], Lundin and Jonsson proposed to use pseudonyms to replace original names. This solution is vulnerable to the *reidentification* attack. A few other methods, e.g., adding "noise" to log files and creating sufficiently complicated pseudonyms, were briefly discussed.

In [12], Biskup and Flegel proposed an adapted version of Shamir's secret sharing scheme [59] to provide better protection against reidentification. Instead of directly mapping a real identity to a pseudonym [48, 49], the identity is mapped to a set of shares of a unique secret. The basic idea of this scheme is to have a pseudonymizer, acting as representative of the anonymity group of all its clients, and split an identifying feature id_{i_g} into as many pseudonyms as are needed to pseudonymize audit records containing id_{i_g}. Given any t_g pseudonyms of id_{i_g} taken from pseudonymous audit records, id_{i_g} can be recovered. In this scheme, the audit service is under the primary control of the Personal data protection officical (PPO) who is fully trusted by the user. Upon a special request, the PPO is able to perform the reidentification. In a later work [28], Flegel discussed about pseudonymizing Unix log files, again, with the help of a PPO. The PPO is trusted both by the users to protect their pseudonymity and by the site security officer(s) (SSO) to ensure accountability in the face of a security incident.

In an important work in the field of privacy-preserving alert sharing [46], Lincoln et al. first investigated several types of potential attacks (e.g., dictionary attacks, probe-response attacks, and alert flooding), and then proposed to use hash function to anonymize identity information. More specifically, in their scheme, standard hash functions and keyed-hash functions are used for internal and external IP addresses, respectively. As a result of balancing privacy and utility, equality comparisons are restricted to external IP addresses in this scheme, and thus this scheme is effective in detecting high-volume events. On the other hand, there is no privacy guaranteed for a participant's IP addresses if they appear in the alert reported by another participant. In addition, the semantics of alert attributes is destroyed due to the usage of hash functions, and thus certain high-level correlation may not be possible. Besides hash-based solutions, Lincoln et al. proposed a few further privacy protection methods, e.g., re-keying by the repository and randomized hot list thresholds, based on the assumption that the alert repository is fully-trusted. Generally, the scheme is effective in detecting simple high-volume events.

In [35,47], a distributed collaborative IDS was proposed. The basic idea is to use bloom filters [13] to built watchlists so that original IP addresses and port numbers in the alerts are anonymized. This method is suitable for matching the value of single property, e.g., suspicious addresses, and can greatly improve the efficiency of large-scale detections in this case. Unlike other approaches that focus on header-based alert data (e.g., IP addresses and port numbers), Parekh et al. combined this bloom filter-based solution with two types of payload-based anomaly sensors that were developed at Columbia University [74–76] to investigate the possibility of performing privacy-preserving payload-based correlation [55]. Before correlating network traffic, the anomaly sensors need to be trained. In addition, this scheme aims at identifying non-semantic commonality and is unsuitable for the scenarios that more complicated correlations [51] are required and the correlator is not fully trusted.

A research issue that is very similar to privacy-preserving intrusion detection/alert sharing is *packet trace anonymization*. Research works in both fields aim at sharing data that are crucial to security or network analysis, while satisfying certain privacy requirement at the same time. The main difference lies in the object of the problem. The object of packet trace anonymization is raw network traffic, while the object of privacy-preserving intrusion detection/alert sharing is (security) application-level data extracted from either network traffic or host-based events. Generally, almost the same set of anonymization techniques, e.g., random permutation and hashing, can be applied to both problems. Current solutions for packet trace anonymization [53, 54] assume the existence of a fully-trusted administrator which is responsible for anonymizing packet traces before publishing them.

Prefix-preserving IP address anonymization [27,80] is one of anonymization techniques that have drawn much attention. It aims at preserving the prefix relationship among IP addresses after shared data are anonymized, and has been used in different contexts [53,65, 66].

Slagell et al. improved the original prefix-preserving IP address anonymization tool implemented in the Crypto-PAN project [27, 80] with an efficient passphrase-based key generation algorithm for the purpose of ensuring the prefix-preserving property while sharing network logs [65]. Afterwards, they developed a tool that supports multiple levels of anonymization so as to provide better trade-offs between security and utility [66]. In [64], Slagell et al. designed an anonymization framework for computer and network logs, and developed an anonymization tool called as FLAIM. This tool includes the implementation of most anonymization algorithms proposed till then. Instead of proposing new anonymization techniques, the key contributions of this work is two-fold. One is to allow the user to specify an anonymity policy that can make use of a variety of anonymization algorithms at run-time. The other is to separate parsing from I/O to allow third parties to add support for additional logs.

In a recent work, Lee et al. [45] proposed a framework for privacy-preserving interdomain auditing. There are three types of entities in this framework: *organizations*, *audit groups*, and *auditors*. Organizations that have a certain degree of trust relationship form an audit group, and then run a group key management scheme so as to share a secret. A paid or contracted auditor will execute a program that all members of an audit group agree to analyze collective audit records. Making use of this shared secret, the authors presented a set of obfuscation methods, two of which (i.e., local greater-then relation and local blinded summation) are new in the field of privacy-preserving log/alert sharing. This framework assumes that "members of the audit group trust one another enough to maintain any group secrets necessary for processing audit records sent to the auditor"[1]. Therefore, it is ideal for the cases such as branches of a large company, but is unsuitable for the scenarios that the sensitivity of shared data is higher than the mutual trusts among the members [52]. Fairness is not provided in this design, since all the alerts generated by the program will be sent to all the members. the auditor returns the Allman et al. [6] also proposed a scheme

[1]Note that, similarly, in our design we also assume that contributors share a secret. However, the major difference is that such a secret is not used to anonymize the whole shared records/alerts but only to perform keyed-hashing on IP addresses.

for the detection of coordinated attacks upon cross-organization information sharing. However, their scheme assumes that the detectives, which are equivalent to the correlator in our scheme, are fully trustworthy.

In [63], Shmatikov and Wang undertook a deep analysis on probe-response attacks against collaborative intrusion detection systems (CIDS). Their work aims at mitigating the effectiveness of probe-response attacks in terms of detecting epidemic phenomena, i.e., high-volume events. The basic idea is that, by employing a fully decentralized structure, the adversaries has a very low chance of observing rare events. They designed a gossip-based architecture for CIDS, and quantitatively analyzed the basic tradeoff between usability (i.e., a CIDSąŕ ability to detect attacks) and privacy (i.e., the security of individual monitors against probe-response attacks).

Research work in *privacy-preserving data mining* and *database obfuscation* [5, 19, 26] is relevant to our work, but differ in a few aspects. Most privacy-preserving database obfuscation techniques require a centralized trusted party to determine the released view based on privacy requirements, e.g., k-anonymity [57]. Privacy-preserving data mining techniques were designed to discover statistic properties over datasets without compromising privacy of an individual contributor. However, they are not suitable for exact matching of a specific string like IP address or malicious code pattern, while ensuring privacy and data confidentiality (i.e., the string itself) at the same time.

Despite the strong privacy protection provided by *secure multiparty computation (SMC)* [24], its usage is limited due to the poor efficiency [46].

There has been extensive research in *anonymous communications* [61], which aims at preventing adversaries from tracing back to the source or the destination of sniffed communications. It is complementary to our work.

2.2.2 Privacy-Preserving Alert Correlation

The focus of this thesis is privacy-preserving alert correlation. Both template-based alert correlation and condition-oriented alert correlation are built on dedicated *matching* operations. To the best of our knowledge, the privacy-preserving alert correlation schemes proposed by Xu and Ning [78, 79] are the first and only known works that can perform complicated correlations [51]. They are based on the idea of concept hierarchies, which generalize sensitive alert attributes to high-level concepts to introduce uncertainty. Their approach consists of two stages: alert sanitization and sanitized alert correlation. The first stage is guided by (differential) entropy. Due to the generalization, exact matching between sanitized attributes is not possible. To achieve correlation, matching based on a casual relation is employed instead, which may lead to false positives. In addition, an aggregation algorithm is presented to improve the quality of alert correlation graphs, at the cost of introducing a higher false negative. In a later work [79], Xu and Ning proposed three new schemes based on the idea of injecting artificial alerts and/or randomization.

2.3 The Intrusion Detection Alert Correlation Scheme Proposed by Ning et al.

In this section, we first review the alert correlation scheme proposed by Ning et al. [51], and then discuss the possible privacy breaches in this scheme. The basic idea of their approach is that, each alert is treated separately, and needed conditions to trigger this attack and possible conditions caused by this attack are considered.

2.3.1 The Architecture of the Alert Correlation Scheme Proposed by Ning et al.

In this scheme, all alerts are extended with pre-conditions and post-conditions and the actual correlation relies on both the alert itself and extended conditions. Figure 2 shows the architecture of the intrusion alert correlator introduced in Ning et al.'s scheme, excluding the components for visualizing correlation results.

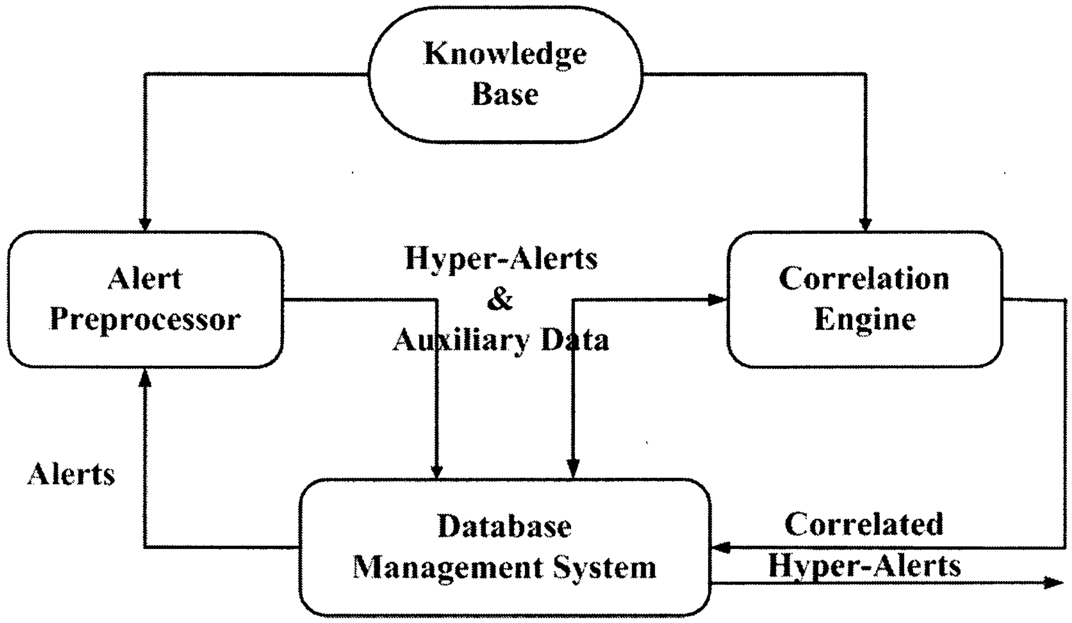


Figure 2: Intrusion Detection Alerts Correlator Proposed by Ning et al.

Their scheme is based on the idea of correlating the preconditions and consequences of individual alerts. In order to perform the correlation, Ning et al. introduced the notation of *Hyper-alert Type (HT)* and *Hyper-alert Instance (HI)*. A HT is defined as a triplet (*fact*, *prerequisite*, *consequence*), where *fact* consists of all attribute names involved in the description of *prerequisite* and *consequence*. A HI is the result of replacing components of the triplet with the corresponding values from a real alert. If the consequence of an earlier hyper-alert A_1 makes one of the prerequisite of the latter one A_2 True, we say A_1 *prepares* for A_2.

In this architecture, *knowledge base* stores the information about all HTs and implication relationship between predicates (i.e., prepare-for relation), and *database management system* stores original alerts collected from IDSs at the beginning of the correlation process. In the first step, *alert preprocessor* converts original/raw alerts received from the database into hyper-alerts and auxiliary data using the HT information from the knowledge base. Afterwards, the hyper-alerts and auxiliary data are stored at the database and then are feed to *correlation engine*. Together with prepare-for relation provided by the knowledge base, the correlation engine can correlate alerts. Finally, the results of correlation are sent back to the database for further processing, e.g., visualization.

2.3.2 Example 1

Consider four raw alerts captured by a RealSecure Network Sensor [39] in Table 1. Each raw alert has eight attributes, i.e., *AlertID*, *AlertType*, *Begin_time*, *End_time*, *DestIPAddress*, *DestPort*, *SrcIPAddress* and *SrcPort*. Among them, *AlertID* is a unique identity assigned by the local sensor when the alerts are generated. The first record indicates that a possible attacker, whose IP address is 202.077.162.213, initialized a *Sadmind_Amslverify_overflow* instance from port 56262 against a target machine with IP address 172.016.112.050 to port 32773 at 04:53:26 AM on 10/11/2001.

Table 1: Raw Alert Records

AlertID(Local)	AlerType	Begin_time	End_time	DestIPAddress	DestPort	SrcIPAddress	SrcPort
63963	Sadmind_Amslverify_overflow	10/11/2001 4:53:26 AM	10/11/2001 4:53:26 AM	172.016.112.050	32773	202.077.162.213	56262
63997	FTP_Put	10/11/2001 5:13:05 AM	10/11/2001 5:13:05 AM	172.016.112.050	32773	202.077.162.213	60569
64163	Stream_Dos	09/11/2001 4:58:07 PM	09/11/2001 4:58:07 PM	131.084.001.031	26341	009.145.123.135	24661
64072	Mstream_Zombie	09/11/2001 4:36:47 PM	09/11/2001 4:36:47 PM	172.016.115.020	9325	172.016.112.050	33345

In the knowledge base, the HT of *Sadmind_Amslverify_overflow* is defined as:

$$Sadmind\_Amslverfiy\_overflow =$$
$$(\{DestIP\},$$
$$VulnerableSadmind(DestIP) \land OSSolaris(DestIP),$$
$$\{GainAccess(DestIP)\})$$

and the HI of the first record is generated as:

$$Sadmind\_Amslverfiy\_overflow =$$
$$(\{172.016.112.050\},$$
$$VulnerableSadmind(172.016.112.050) \land OSSolaris(172.016.112.050),$$
$$\underline{\{GainAccess(172.016.112.050)\}})$$

Similarly, the HT of *FTP_Put* is:

$$FTP\_Put =$$
$$(\{DestIP, DestPort\},$$
$$ExistService(DestIP, DestPort) \land GainAccess(DestIP),$$
$$\{SystemCompromised(DestIP)\})$$

and the HI of the second record is generated as:

$$\begin{aligned}&FTP\_Put = \\ &(\{172.016.112.050, 32773\}, \\ &ExistService(172.016.112.050, 32773) \wedge \underline{GainAccess(172.016.112.050)}, \\ &\{SystemCompromised(172.016.112.050)\})\end{aligned}$$

As highlighted in previous expressions, i.e., the unlined parts, the consequence of the HI corresponding to the first record, i.e., $\{GainAccess(172.016.112.050)\}$, is one of the prerequisites of the HI of the second record. Hence, these two alerts are correlated in Ning et al.'s scheme.

2.3.3 Privacy in Ning et al.'s Alert Correlation Scheme

Attackers have various approaches to defeat contributors' privacy against the above correlator. For example, by accessing DBMS or public correlated results. This is due to the design that plaintexts are stored or published. More specifically, by knowing the *AlertType* and *SrcIPAddress* or *DestIPAddress*, an attack can easily link these attributes to learn the possible vulnerabilities on a particular machine and to launch a direct attack. To keep the content of alerts secret while preserving the utility of correlation algorithms is one of intents of this work.

Another possible approach to deduce a contributor's private information is by acting as eavesdroppers, observing and decoding certain information (e.g. IP addresses) from the network traffic package between communicating computers, and tracing the source of the connection, further compromising the identity.

2.4 Predicate Privacy in PEKS

Public-key Encryption with Keyword Search (PEKS) introduced by Boneh et al. [15] is the first practical asymmetric searchable encryption scheme, as well as the first predicate encryption scheme. In this section, we give background knowledge of PEKS and present related predicate privacy issues.

2.4.1 A Brief Review of PEKS

PEKS is originally designed for the purpose of intelligent email routing. For example, as shown in Figure 3, a user R may receive emails through different devices, e.g., a PDA or a desktop at the office. Hence, she/he may want to selectively forward emails with certain keywords to a specific device, e.g., emails that contain the word "agenda" are forwarded to the PDA. To protect data confidentiality, emails are encrypted at the sender side. Hence, the mail server G has no access to the content of emails. To delegate G the capability of performing selective forwarding, however, G is assigned a set of trapdoors that are corresponding to keywords that might be used for searching at a later time. For a keyword x, the corresponding trapdoor $t(x)$ is generated from the master secret held only by R and is used to define a predicate p. Upon receiving a ciphertext, G can verify whether the corresponding plaintext is equal to x based on the predicate.

As shown in Figure 3, there are three types of entities in the PEKS scheme [15]: *receiver* R, *sender*, and *server* G. PEKS consists of the following four procedures:

1. $KeyGen(s)$: Takes as input a security parameter s, the receiver R generates a PEKS public/private-key pair, i.e., $\{A_{pub}, A_{priv}\}$, as well as other public parameters;

2. $Trapdoor(A_{priv}, x)$: given the private key A_{priv} and a keyword x, the receiver R produces a corresponding trapdoor T_x;

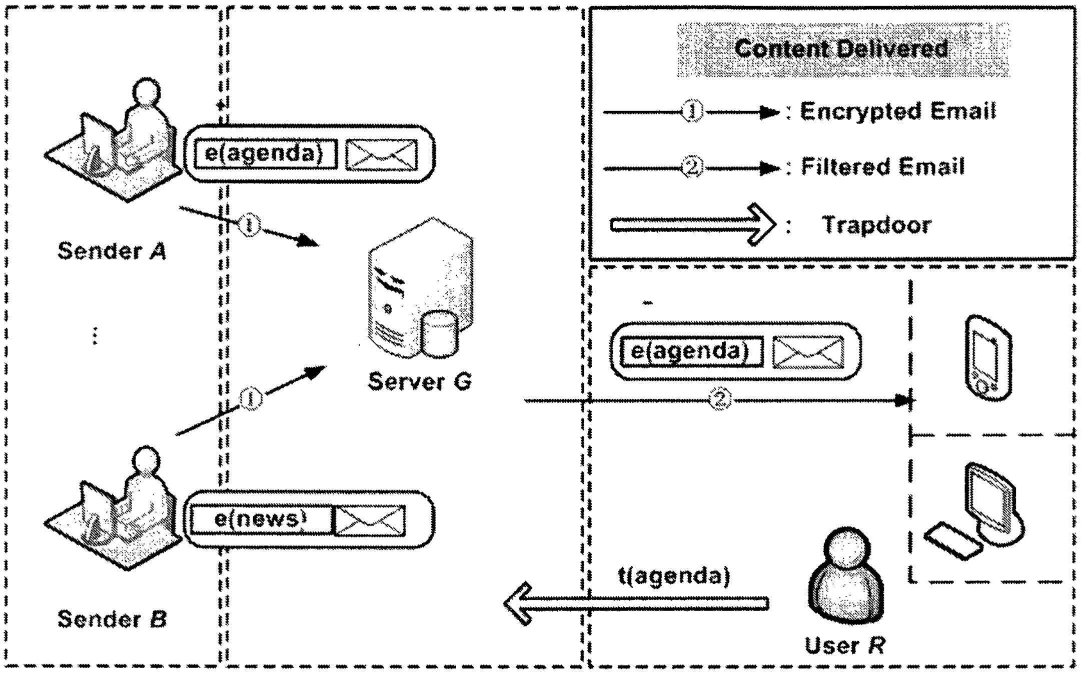


Figure 3: Public Key Encryption with Keyword Search Framework

3. $PEKS(A_{pub},\ x)$: given the receiver's public-key (i.e., A_{pub}) and a keyword x, a *sender* generates the PEKS ciphertext of a message (e.g., an email) to be sent to R, which is denoted as S;

4. $Test(S, T_x)$: given the received PEKS ciphertext $S = PEKS(A_{pub},\ x')$ and a trapdoor $T_x = Trapdoor(A_{priv},\ x)$, the *server* G outputs 'yes' if $x = x'$ and 'no' otherwise.

An instantiated construction of PEKS is based on a bilinear map of elliptic curves. It uses two cyclic groups G_1, G_2 of the same prime order p and a symmetric bilinear map $e : G_1 \times G_1 \rightarrow G_2$ between them. If we use a multiplicative notation to describe the operation in G_1 and G_2, e has following properties:

- Bilinear: $e(g^x, g^y) = e(g, g)^{xy}$ for all integers $x, y \in [1, p]$, $g \in G_1$;

- Non-degenerate: $e(g,g) \neq 1$ and if g is a generator of G_1 then $e(g,g)$ is a generator of G_2;
- Computable: There is a polynomial time algorithm to compute $e(g,h) \in G_2$ for all $g,h \in G_1$.

The security of PEKS is based on the assumption of Elliptic Curve DLP, which is believed to be intractable for certain carefully chosen groups including the group that is formed by the points on an elliptic curve defined over a finite field. More specifically, given two points on an elliptic curve, g and g^x, where x is a scalar, it is computationally infeasible to obtain x, if x is sufficiently large.

2.4.2 Predicate Privacy in PEKS

Most previous works on predicate encryption concentrate on *plaintext privacy*, i.e., the property that ciphertexts reveal no information about the encrypted data to any party without the private key other than what is inherently revealed by the trapdoors. However, researchers also identified a few other security/privacy issues relevant to PEKS [3,8,18,60]. One major concern is to limit the delegate's capability of keyword searching within a certain time frame [3,8]. Another important concern is that PEKS is subject to offline keyword guessing attacks firstly identified by Byun et al [18]. Later, Shen, Shi, and Waters formalized the second concern and introduced the notion of *predicate privacy* [60], i.e., the property that $t(x)$ reveals no information about the encoded predicate p. They also proposed a predicate encryption scheme that can achieve both plaintext privacy and predicate privacy in the symmetric-key settings. Moreover, Shen, Shi, and Waters claimed that it is inherently impossible to achieve predicate privacy in the public-key setting, such as PEKS. Interesting though, several researchers had actually proposed a few solutions to this problem [3,8]. However, Shen, Shi, and Waters's claim may be based on an implicit assumption that the proposed solution should not conflict with one of the aims of PEKS, i.e., making keyword search possible without interaction between the sender and receiver, which was indicated by Baek et al [8]. Such an assumption is definitely reasonable, since in practice the size of potential senders could be a huge number. Moreover, the proposed solutions [3, 8] require to share some secret, in the form of either a set of public keys of the receiver [3] or the method of refreshing keywords [3, 8], between senders and the receiver. Considering the huge number of potential senders, the overhead of synchronizing the secret and protecting it from disclosure is overwhelming.

Formally, the predicate p of the equality test in PEKS can be defined as $p(e(x), t(x)) = 1$, in which x is the keyword, $e(x)$ is the encryption of x, and $t(x)$ is the trapdoor derived from x and the private key held only by R. In reality, the mail server G is usually considered as a semi-trusted entity, which is honest-but-curious [33]. Since the public-key encryption function does not require a secret key, G can encrypt any plaintext of her choice and then evaluate the resulting ciphertext with the trapdoors assigned by R. By verifying whether the resulting ciphertext satisfies the predicate associating with a trapdoor, G learns whether the chosen plaintext is equal to the keyword that is corresponding to the trapdoor. In other words, the mail server G can launch an attack similar to brute-force password attacks. In particular, PEKS is especially fragile to this type of attacks in those applications, in which there exists a small set of keywords that are frequently used, such as "Urgent" and "Classified". In this paper, we refer this type of attacks as *brute-force guessing attacks.*

Our solution to brute-force guessing attacks are based on two basic ideas. One is to introduce randomness into the procedure of generating trapdoors so as to avoid the deterministic one-to-one mapping between a searching keyword and the corresponding trapdoor. It is also the underlying idea of previous solutions [3,8]. The other is to limit the knowledge of the secret introducing randomness to only one or a small set of entities. Through creating random instances of keywords, the actual plaintexts used in the generation of trapdoors are known only to the entities that know the secret used in the randomization. Hence, in terms of the intelligent email routing, the semi-trusted mail server G cannot launch brute-force guessing attacks any more. On the other hand, by limiting the holders of the secret, the overhead of synchronizing and protecting the secret is much smaller.

Besides brute-force guessing attacks, an alternative attack that may be launched by the mail server is to make use of external knowledge about the statistical distribution of keywords to identify the relation between a trapdoor/predicate and a keyword. We call it as a *statistical guessing attack*. For example, in an application the probabilities that a keyword x and any other keyword are matched are 20% and no more than 10%. In such a case, the mail server can easily deduce the trapdoor corresponding to x by simply counting the number of times that each trapdoor is matched. Note that, the method of refreshing keywords by appending the time period to the keyword before the encryption is still subject to this type of attacks, because it does not change the statistical distribution of keywords within the same time period. Informally, the idea of our solution to statistical guessing attacks is to spread out the statistical distribution of keywords. In the previous example, it is much more difficult for the mail server to guess when the probabilities that a trapdoor mapped to keyword x and a trapdoor mapped to any other keyword are matched are 2% and no more than 1%, given that the same number of keyword matching events are observed. Apparently, when the mail server observes a sufficient number of events, it can still figure out the mappings between trapdoors and keywords. Hence, we have to refresh such mappings before it happens. In this paper, we provide detailed analysis on identifying an appropriate balance between privacy and efficiency. In Chapter 6, we proposed the PEKSrand scheme to provide strong privacy protection in PEKS. PEKSrand has two variants: *PEKSrand-BG* and *PEKSrand-SG*. Both variants are robustly against brute-force guessing attacks, and thus can ensure predicate privacy in the scenarios where the statistical distribution of keywords to be searched is unknown. Compared to PEKSrand-BG, PEKSrand-SG can further mitigate statistical guessing attacks at the cost of the storage overhead on the delegate, e.g., the mail server in the intelligent email routing application. According to our analysis and experimental results, both schemes introduce negligible additional communication and computation overheads and can be smoothly deployed in existing systems.

Chapter 3

The Framework of the TEIRESIAS Protocol

In this chapter, we describe the system and adversary models assumed in the design of the TEIRESIAS protocol. Afterwards, the design goals and the basic idea of our design are presented, followed by an outline of the system framework of TEIRESIAS.

3.1 System Model and Adversary Model

Our design aims at providing privacy-preserving alert correlation and retrieval under the cross-domain setting. We assume that there are a group of domains/organizations that want to share and correlate their alerts, and they may not have a full trust on each other.

We also assume that there is an offline fully trusted authority, denoted as $\mathbb{O}$, which is responsible for the selection of system-wide parameters and the generation of security tokens. Note that, to avoid the potential risk that the fully trusted entity is compromised, a major difference of our design from many previous schemes is that it does not require $\mathbb{O}$ to be online, since the tasks of alert correlation and retrieval do not involve $\mathbb{O}$, once the system has been set up. Instead, the task of alert correlation and the task of retrieving correlation results are handled by a correlator and a data server that stores the correlation results, respectively. We assume that, both the correlator and the data server are semi-trusted, i.e., honest-but-curious [33].

We assumed that, the adversary could be an insider (i.e., a domain participating in alert sharing), or an outsider, or both. In the last case, the outsider(s) collude(s) with one or a few insiders so as to break the system. We further assume that, the insiders restrain themselves from launching active attacks. Such an assumption is reasonable, since active attacks are much more easier to be detected and traced and the insiders may want to avoid relevant legal issues. In addition, the exposure of insiders may greatly reduce the effectiveness of future attacks.

An overall solution of privacy-preserving alert sharing should also include privacy protection during alert collection, e.g., employing an anonymous routing protocol like Tor [23]. The discussion about the design of such an anonymous communication protocol is beyond the scope of this paper. It is complementary to our work.

3.2 Design Goals

A scheme that aims at providing secure and privacy-preserving cross-domain alert correlation and retrieval should satisfy the following requirements:

- **Data Confidentiality** The content (e.g., alert type and source/destination IP addresses) of any alert shared should be protected from both insiders and outsiders, except the source of this alert.
- **Unlinkability** Both insiders and outsiders can neither link an alert with its source, i.e., the entity that shares this alert, nor link alerts from the same source.
- **Fairness** An entity that participates in the cross-domain alert correlation system can only retrieve those correlation results that are pertinent to the alerts that it shares.

- **Usability** The results of alert correlation upon transformed alerts stored on the correlator should be the same as, or at least with few differences from, that upon original alerts.

3.3 Building Blocks and Basic Idea of Our Design

To ensure data confidentiality and unlinkability, an easy way is to encrypt the data to be protected. However, simple encryption destroys the structure of the data, and thus the usability of such data is extremely limited. Therefore, we adopt encrypted keyword search algorithms as candidates in this thesis. In the past several years, there has been many research works on searchahle encryption, which aims at balancing the security and usability of the outsourced data.

The schemes for searchable encryption can be divided into two categories: symmetric-key encryption with keyword search (SEKS) [32, 68] and public-key encryption with keyword search (PEKS) [15].

In a typical application of SEKS, e.g., Song et al.'s scheme [68], a user U outsources her/his data in the encrypted format to a remote server S. When user U wants to search for a particular keyword on the outsourced data, she/he needs to inform the remote server the corresponding searching key. The remote server will return the result, if any, to the user. However, it has no idea abcut the keyword or the content (i.e., the plaintext) of the searching result. If we ignore the process that U discloses a searching key corresponding to a keyword to S, SEKS can be simplified into two steps: 1) U outsources data to S; 2) S searches on the encrypted data and returns the results to U. We define it as the *U(ser)-S(erver)-U(ser)* model, **USU** for short. Note that, in the USU model, the scope of a searching operation is bounded by the data owner.

In the original design of PEKS [1], upon receiving security tokens from user U that are

[1]A brief review is presented in Section 2.4.1

corresponding to certain keywords, the mail server is capable of filtering those emails containing a specific keyword and then forwarding them to a specific device. However, the mail server has no idea about the exact keyword corresponding to the security token that it possesses. If we ignore the process that U discloses a searching key corresponding to a keyword to server S, PEKS can be simplified into two steps: 1) multiple users send messages to U, which are stored by S; 2) S searches on the encrypted data and forwards the results to U. We define it as the *M(ulti-user)-S(erver)-U(ser)* model, **MSU** for short. Note that, in the MSU model, the scope of a searching operation is bounded by the intended recipient of the data.

Apparently, these two categories of searchable encryption schemes aim at different types of applications, and thus have distinct system models. In terms of the application on which our work concentrates, we observe that, alert correlation and alert retrieval fall into the MSU model and the USU model, respectively. Therefore, the basic idea of our design is to employ PKES for providing privacy-preserving alert correlation without requiring an online trusted authority and to employ SEKS for the purpose of privacy-preserving retrieval. Note that, the 'S' in MSU and USU in our design are not the same entity. Instead, they are corresponding to the correlator and the data server, respectively. To the best of our knowledge, hence, there is no previous work that employs SEKS and PEKS in the same system or application.

3.4 System Framework and Outline of TEIRESIAS

The proposed system framework is shown in Figure 4. The TEIRESIAS protocol uses the following five procedures: *setup*, *anonymization & R (etrievable)-token concatenating*, *alert correlation*, *report retrieval* and *advanced analysis* (optional). We now present an outline of the operations of TEIRESIAS in the basic scheme. (The advanced scheme is described in Section 4.3. In this scheme, the setup, anonymization and alert correlation operate differently.)

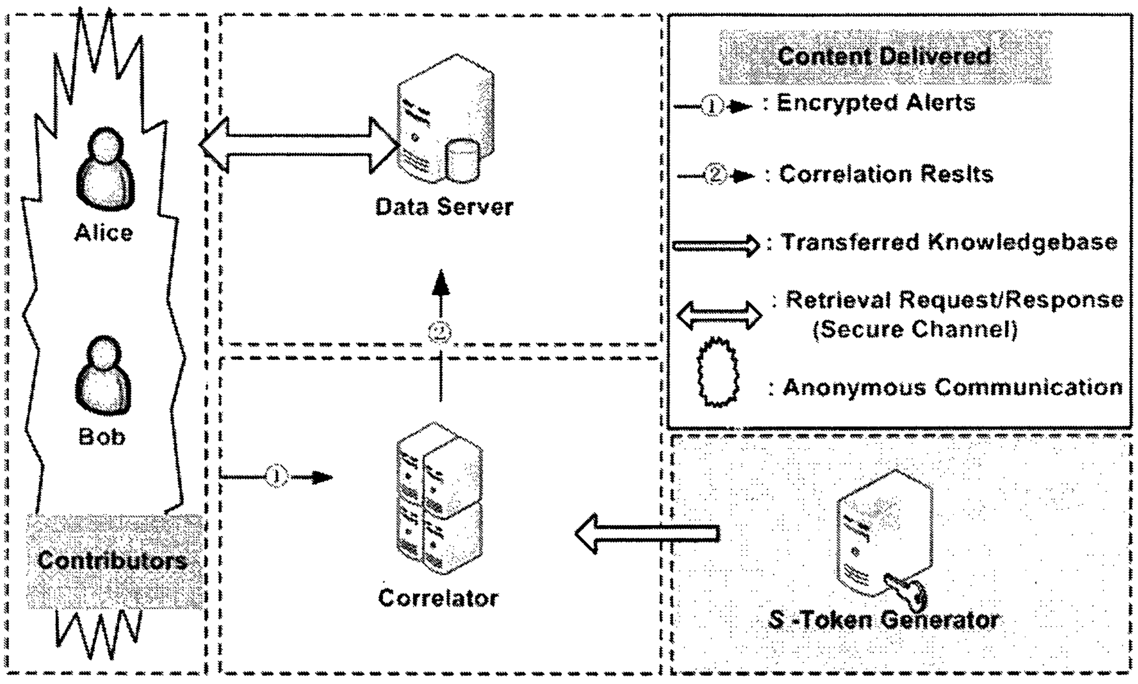


Figure 4: Proposed System Framework of TEIRESIAS

- During *setup* phase, the S-token generator, which plays the role of $\mathbb{O}$ in the system model, generates a network wide public /secret key pair and other public information, e.g., the security parameters chosen. In addition, S-token generator replaces sensitive keywords in the knowledgebase by its corresponding trapdoors. Subsequently, transformed knowledgebase is delivered to the correlator. Once transformed knowledgebase and public parameters are released, S-token generator may remain offline until the next setup operation.

- *Anonymization* & *R-token concatenating* are both done by each contributor, who anonymizes every *AlertType* by using S-token generator's public key and a random number chosen by the contributor. Then, the plaintext of *SrcIPAddress* and *DestIPAddress* are replaced by the corresponding keyed-hash results. After anonymization, every anonymized alert is concatenated with an R-token, which is generated from a symmetric key and a secret number kept by the contributor.

- *Alert correlation* correlates encrypted alerts based on the transformed knowledge-base.

- *Report retrieval* addresses the fairness issue by allowing a contributor to query related correlation results back. Through analyzing a retrieved report, two goals can achieve: first, understand both local and cross-domain correlation results, and second, be aware of probe-response and collusion attacks.

- *Advanced analysis* is an optional procedure, which is only provided to fully-trusted and highly privileged analysts to reconstruct detailed attack scenarios.

Chapter 4

The TEIRESIAS Protocol

In this chapter, firstly, the procedures of TEIRESIAS in the basic and advanced schemes are presented. Then, we focus on several previous attacks against privacy-preserving information sharing and discuss how security and privacy-related properties are achieved in TEIRESIAS. Finally, we identify a particular collusion attack against the TEIRESIAS protocol and propose the countermeasure.

4.1 Notations and Terminologies

- **Bilinear Groups and Bilinear Map** Let G_1,G_2 be cyclic groups whose order is a prime p and let e: $G_1 \times G_1 \rightarrow G_2$ be a symmetric bilinear pairing between G_1,G_2 with following properties:

  1. *Bilinearility*: $e(g^x, g^y) = e(g, g)^{xy}$ for all integers $x, y \in [1, p]$, $g \in G_1$;
  2. *Non-degeneracy*: $e(g, g) \neq 1$ and if g is a generator of G_1 then $e(g, g)$ is a generator of G_2;
  3. *Computability*: there is an polynomial time algorithm to compute $e(g, h) \in G_2$ for all $g, h \in G_1$.

- **Hash Functions** $H_1 : \{0,1\}^* \rightarrow G_1$ is a hash function which maps any variable-length message to a group member in G_1; $H_2 : G_2 \rightarrow \{0,1\}^{\log p}$ is another hash function which maps a group member in G_2 to a binary string whose length is $\log p$.

- **Keyed-hash Function** HMAC represents a system wide keyed-hash function involving a cryptographic hash function, such as MD5 or SHA-1, in combination with a uniform secret key shared among all contributors. One construction of such keyed-hash function is proposed in [9].

- **Pseudorandom Number / Generator / Functions** Define $X = \{0,1\}^{n-m}$, $Y = \{0,1\}^m$ and a key set κ, then

  1. A pseudorandom number B_λ is outputted by the pseudorandom generator $G : \kappa_G \rightarrow X^\lambda$ for a specified λ, which is the total number of R-token.

  2. If $\kappa \times \chi \rightarrow \gamma$ is used to represent pseudorandom functions, we initialize two pseudorandom functions $F : \kappa_F \times X \rightarrow Y$ and $f : \kappa_f \times \{0,1\}^* \rightarrow \kappa_f$ keyed independently. We write $F_k(x)$ and $f_k(x)$ for the results of feeding x to F and f with $k \in \kappa$, respectively.

- **Deterministic Encryption** E Suppose N is a secret number, the computation of $E(N) = \{0,1\}^n$ depends only on N. For instance, ECB encryption of N using some block ciphers is an implement of E.

- **Symbols** In the description of TEIRESIAS, for any two objects (i.e., alerts, alert attributes or ciphertexts): x and y, we write $x||y$ for the concatenation of x and y; $x \oplus y$ for the bitwise XOR of x and y; $x \rightarrow y$ for the discovered the relation "Alert x prepare for alert y".

  I (resp. i) is the local (respectively, global) ID number assigned by the contributor (resp. correlator) to the alert a.

4.2 The Basic Scheme

In this section, we present the procedures of TEIRESIAS in the basic scheme, followed by a concrete example to illustrate the details.

Generally, the S-token generator uses H_1 to generate S-tokens from *AlertType* in the knowledgebase by mapping these keywords into the group G_1 so as to anonymize the knowledgebase. Each contributor employs H_1, H_2, HMAC and e to anonymize sensitive attributes (i.e., *AlertType*, *SrcIPaddress* and *DestIPaddress*) in the raw alerts. Meanwhile, each contributor also computes the bitwise exclusive XOR of $E(N)$ with an artificial token T to construct one R-token, C_N, for the retrieval function.

4.2.1 Procedures of The Basic Scheme

In the following description of the procedures, by default, we assume all system-wide parameters are publicly known once they are set up.

- **Setup** The input security parameter determines the size, p, of the groups G_1 and G_2. S-token generator picks a random $\alpha \in Z_p^*$ and a public generator g of G_1. Then outputs public key pair $A_{pub} = [g, h = g^{\alpha}]$ and $A_{priv} = \alpha$;

  The S-token for each keyword W_{kat}, a *AlertType* in the knowledgebase, is generated as a trapdoor $T(W_{kat}) = H_1(W_{kat})^{\alpha} \in G_1$. After replacing all W_{kat} with the corresponding S-tokens, the anonymized knowledgebase is delivered to the correlator through a secure channel built between S-token generator and correlator;

  Meanwhile, all potential contributors authenticate themselves to an authentication sever in order to get the uniform secret key, which is used to calculate HMAC of *SrcIPAddress* and *DestIPAddress*. Such an authentication service can be achieved by deploying Kerberos [50] for instance.

- **Anonymization & R-token Concatenating** For each sensitive *AlertType*, W_{rat} in the raw alert a, the contributor first picks a random $r \in Z_p^*$, and computes $t = e(H_1(W_{rat}), h^r) \in G_2$, then outputs PEKS ciphertext $PEKS(W_{rat}) = [g^r, H_2(t)]$ to replace W_{rat}. For each sensitive *SrcIPAddress* or *DestIPAddress* (i.e., W_{SrcIP} or W_{DestIP}) in the raw alert a, the plaintext is replaced by the corresponding keyed-hash result: $HMAC(W_{SrcIP})$ or $HMAC(W_{DestIP})$.

  If raw alert a is formated as following:

  a = I || W_{rat} || *Begin_time* || *End_time* || W_{DestIP} || *DestPort* || W_{SrcIP} || *SrcPort*

  Then, the anonymized a is:

  a = I || PEKS(W_{rat}) || *Begin_time* || *End_time*
  || HMAC(W_{DestIP}) || *DestPort* || HMAC(W_{SrcIP}) || *SrcPort*

  The following four steps are executed to generate one R-token:

  1. First, encrypts a n-bit concatenation N of a x-bit secret number and a $(n - x)$ bits unique local index I using the encryption algorithm E, and $E(N)$ is still $n-$bit. Then $E(N)$ is split into two parts, $E(N) = E_1(N)||E_2(N)$, where $E_1(N)$ (respectively, $E_2(N)$) denotes the first $n - m$ bits (resp. last m bits) of $E(N)$;

  2. Second, a key $k = f_{k'}(E_1(N))$ is chosen, where k' is chosen uniformly and randomly in κ by the contributor and never revealed;

  3. Third, pseudorandom bits B_λ are generated by the pseudorandom generator G, and B_λ is $(n - m)$ bits;

  4. Finally, takes B_λ and key k to set $T = B_\lambda||F_k(B_\lambda)$, and outputs the R-token $C_N = E(N) \oplus T$.

  After removing the A*lerID(Local) I* in the anonymized alert a, the generated R-token C_N is concatenated with a and formated as following:

$$\begin{aligned} \text{a} = C_N \,\|\, \text{PEKS}(W_{rat}) \,\|\, \textit{Begin\_time} \,\|\, \textit{End\_time} \\ \|\, \text{HMAC}(W_{DestIP}) \,\|\, \textit{DestPort} \,\|\, \text{HMAC}(W_{SrcIP}) \,\|\, \textit{SrcPort} \end{aligned}$$

- **Alert Correlation** To generate the HI of an alert through identifying the corresponding HT in the knowledgebase, Ning et al.'s scheme is based on the *exactly matching* of the HT and the alert's *AlertType*. However, the S-token and PEKS ciphertext generated from a same *AlertType* are no longer exactly matching. For example, $T(FTP\_Put) \neq PEKS(FTP\_Put)$. Thus, in TEIRESIAS, we use Algorithm 1 to generate HIs.

  Afterward, the actual correlation of HIs can be efficiently achieved by using a simple "*SELECT-FROM-WHERE*" SQL query. Finally, by linking correlated alert pairs, different attack scenarios at different detailed levels can be reconstructed through report retrieval and advanced analysis procedures.

  To support report retrieval and advanced analysis, the correlator extracts different information from correlated alert pairs. On the one hand, R-tokens are extracted from correlated alert pairs and sent to the data server. On the other hand, *AlertID (Global)* assigned by the correlator for all incoming alerts are extracted from correlated alert pairs and stored locally. Also note that the paired relationship between each two extracted attributes (i.e., either R-token or *AlertID (Global)*) is preserved in both formats.

- **Report Retrieval** Report retrieval procedure enables contributors to get correlated results which they involved in back, while guaranteeing that no one can retrieve any other contributors' private alerts without her authorization (i.e., the particular retrieval query).

  To construct a retrieval query, the contributor reveals $< E(N), k >$ pairs to the data server, who searches each R-token received from the correlator by checking whether

Algorithm 1 Privacy-Preserving HIs Generation in the Basic Scheme

```
Input: (1) Anonymized alerts $a_i$, in which
           $PEKS(W^i_{rat}) = [g^{r_i}, H_2(t_i)]$, and $t_i = e(H_1(W^i_{rat}), h^{r_i})$;
       (2) HTs with AlertType $T(W^j_{kat}) = H_1(W^j_{kat})^\alpha$.
Output: HIs.
 1: Set $i = 1$;
 2: for alert $a_i \in$ Database do
 3:   Set $j = 1$;
 4:   for $T(W^j_{kat}) \in S$-tokens do
 5:     if $H_2(e(T(W^j_{kat}), g^{r_i})) = H_2(t_i)$ then
 6:       Generate HI from HT with $T(W^j_{kat})$, goto step 11;
 7:     else
 8:       $j = j + 1$;
 9:     end if
10:   end for
11:   $i = i + 1$;
12: end for
13: return 0
```

$C_N \oplus E(N)$ is of the form $S||F_k(S)$ for some S. If so, the data server marks this satisfied R-token (e.g., concatenates a symbol ! with the satisfied R-token as $!||C_N$). Otherwise, encrypts this R-token by another deterministic encryption algorithm $E'(C_N)$ whose secret key is only kept by the data server. Finally, all pairs, which have marked R-tokens, are delivered back to the querying contributor. If none of the R tokens is marked, just returns no alert correlated. In particular, If only interested in some specific alerts, the contributor just reveals those corresponding $< E(N), k >$ pairs.

To eliminate the potential possibility that a requester's identity exposed through the messages in the public network, a secure channel such as SSL [31] is required to set up between each contributor and the data server to transfer retrieval request / response.

Upon receiving returned reports, by XOR first $(n - m)$ bits of marked C_N with B_λ, which only can be generated by the pseudorandom generator G with the original seed, the alert owner can regain $(n - m)$ bits $E_1(N)$, which allows to compute k

and thus recovers the $(n - x)$ bits unique local index number I. Finally, by cross-referring to the raw alerts set, correlated results can be learned. Note that in reports, the deterministic encryption algorithm $E'(C_N)$ assures that the ciphertexts generated from the same R-token are always exactly matching, although R-tokens from other contributors are not allowed to decrypt due to privacy concerns. Thus, the usability of retrieved report is maximized.

- **Advanced analysis (optional)** Since all the ciphertexts could be unsealed, advanced analysis function is only provided to fully-trusted and highly privileged analysts to reconstruct detailed attack scenarios.

  To perform this function, analysts first get all encrypted alerts and *AlertID (Global)* pairs prepared by the correlator, as well as original knowledgebase and corresponding S-toke list held by S-token generator. Then, based on *AlertID (Global)* pairs, analysts extract correlated encrypted alerts and picture their prepare-for relationships. Finally, by *matching* (since decryption function is not supported in TEIRESIAS) *AlertType* attributes in the encrypted alerts with trapdoors in the original S-toke list, analysts can reconstruct detailed attack scenarios for further study.

Table 2: Anonymized Alert Records

AlertID(Global)	R-token	AlertType	Begin_time	End_time	DestIPAddress	DestPort	SrcIPAddress	SrcPort
125	$C_{N_{6,3903}}$	PEKS(Sadmind_Amslverify_overflow)	10/11/2001 4:53:26 AM	10/11/2001 4:53:26 AM	HMAC(172.016.112.050)	32773	HMAC(202.077.162.213)	56262
430	$C_{N_{6,3997}}$	PEKS(FTP_Put)	10/11/2001 5:13:05 AM	10/11/2001 5:13:05 AM	HMAC(172.016.112.050)	32773	HMAC(202.077.162.213)	60569
173	$C_{N_{6,1164}}$	PEKS(Stream_Dos)	09/11/2001 4:58:07 PM	09/11/2001 4:58:07 PM	HMAC(131.084.001.031)	26341	HMAC(009.145.123.135)	24661
334	$C_{N_{6,4072}}$	PEKS(Mstream_Zombie)	09/11/2001 4:36:47 PM	09/11/2001 4:36:47 PM	HMAC(172.016.115.020)	9325	HMAC(172.016.112.050)	33345

4.2.2 Example 2

Table 2 shows the anonymized raw alerts of Table 1. The anonymized HTs in example 1 with S-tokens are:

$T(Sadmind\_Amslverfiy\_overflow) =$

$(\{DestIP\},$

$T(VulnerableSadmind)(DestIP) \wedge T(OSSolaris)(DestIP),$

$\{T(GainAccess)(DestIP)\})$

$T(FTP\_Put) =$

$(\{DestIP, DestPort\},$

$T(ExistService)(DestIP, DestPort) \wedge T(GainAccess)(DestIP),$

$\{T(SystemCompromised)(DestIP)\})$

By executing Algorithm 1, The generated HIs of first two encrypted alerts in Table 2 are:

$PEKS(Sadmind\_Amslverfiy\_overflow) =$

$(\{HMAC(172.016.112.050)\},$

$T(VulnerableSadmind)(HMAC(172.016.112.050)) \wedge$

$T(OSSolaris)(HMAC(172.016.112.050)),$

$\underline{\{T(GainAccess)(HMAC(172.016.112.050))\}})$

$$PEKS(FTP\_Put) =$$
$$(\{HMAC(172.016.112.050), 32773\},$$
$$T(ExistService)(HMAC(172.016.112.050, 32773)) \wedge$$
$$\underline{T(GainAccess)(HMAC(172.016.112.050))},$$
$$\{T(SystemCompromised)(HMAC(172.016.112.050))\})$$

The exactly matching of unlined parts of above two HIs enables the correlator judging that $a_{125} \rightarrow a_{430}$. Similarly, the corresponding knowledgebase also guides the correaltor discovering that four encrypted alerts in Table 2 also meet following two extra relations: $a_{430} \rightarrow a_{334}$ and $a_{334} \rightarrow a_{173}$.

To support report retrieval, the R-token pairs sent to the data server are organized as:

$$\boxed{\begin{array}{c} C_{N_{63963}} \rightarrow C_{N_{63997}} \\ C_{N_{64072}} \rightarrow C_{N_{64163}} \\ C_{N_{63997}} \rightarrow C_{N_{64072}} \end{array}}$$

Suppose in Table 2, alerts with R-tokens $C_{N_{63963}}$, $C_{N_{63997}}$ and $C_{N_{64072}}$ are contributed by a network-based intrusion detection system, Alice. Alert with R-token $C_{N_{64163}}$ is contributed by another intrusion detection system, Bob.

By checking above R-token pairs with following retrieval query from Alice,

$$\boxed{\begin{array}{c} [\, E(N_{63963}),\ k_{63963} = f_k(E_1(N_{63963}))\,] \\ [\, E(N_{63997}),\ k_{63997} = f_k(E_1(N_{63997}))\,] \\ [\, E(N_{64072}),\ k_{64072} = f_k(E_1(N_{64072}))\,] \end{array}}$$

the data sever marks and sends the following report back to Alice:

$$\boxed{\begin{array}{c} !||C_{N_{63963}} \rightarrow\ !||C_{N_{63997}} \\ !||C_{N_{64072}} \rightarrow E'(C_{N_{64163}}) \\ !||C_{N_{63997}} \rightarrow\ !||C_{N_{64072}} \end{array}}$$

It is very convenient for Alice to restore *AlertID(Local)* from the above marked *R*-tokens, and the final report learned by Alice is:

$$\boxed{a'_{63963} \xrightarrow{Prepare-For} a'_{63997} \xrightarrow{Prepare-For} a'_{64072} \xrightarrow{Prepare-For} E'(C_{N_{64163}})}.$$

Further, alice can reconstruct attack scenario which she is involved in as Figure 5 through the above report.

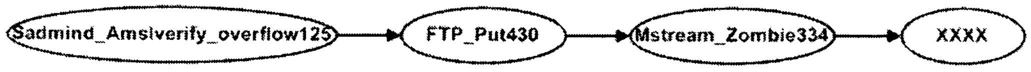


Figure 5: The First Attack Scenario Reconstructed by Report Retrieval

Similarly, the final report learned by Bob is:

$$\boxed{E'(C_{N_{64072}}) \xrightarrow{Prepare-For} a'_{64163}}$$

and the attack scenario learned by Bob is in Figure 6.

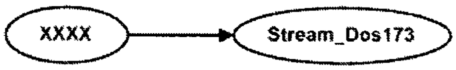


Figure 6: The Second Attack Scenario Reconstructed by Report Retrieval

To support advanced analysis, three locally stored *AlertID (Global)* pairs are formated as:

$$\boxed{\begin{array}{c} 125 \rightarrow 430 \\ 334 \rightarrow 173 \\ 430 \rightarrow 334 \end{array}}$$

Through cross referring to the encrypted alerts in Table 2, the above *AlertID (Global)* pairs enable analysts to identify the following attack relationships among those four alerts:

$$\boxed{a_{125} \xrightarrow{Prepare-For} a_{430} \xrightarrow{Prepare-For} a_{334} \xrightarrow{Prepare-For} a_{173}}$$

Finally, the detailed attack scenario can be constructed as shown in Figure 7.

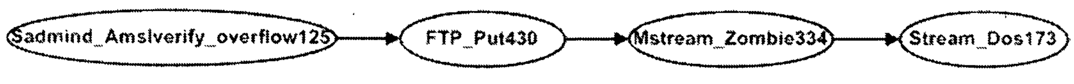


Figure 7: The Attack Scenario Reconstructed by Advanced Analysis

4.3 The Advanced Scheme

An ideal keyed-hash function used in the basic scheme to anonymize IP addresses is able to withstand all known types of cryptanalytic attacks, however, the security and privacy of IP attributes are totally relied on the secrecy of the shared uniform key to generate keyed-hash values. In certain scenarios, especially in the cross-domain intrusion detection alert correlation setting, the confidentiality of the uniform key can not be guaranteed. For example, the key may be leaked through a contributor which is a undetectable insider. At this point, the privacy of IP attribute is compromised in the basic scheme. Nevertheless, as our last defense, the basic scheme still preserves the unlinkability between the *AlertType* and *SrcIPAddress* (or *DestIPAddress*) in any one of the anonymized alerts, which baffles a attacker to figure out the specific security vulnerability associated with that IP address.

Anonymizing IP address also drew much attention in recent works. However, the proposed schemes are all infeasible in our case. For instance, using a uniform hash function and different keyed-hash functions to digest external and internal IP addresses respectively [46] may restrict equality test of internal IP addresses; Using concept hierarchy approach to generalize detailed values into a higher level value [78] introduces uncertainty and fails to perform delicate equality test; Using prefix-preserving IP address anonymization [27, 80] can only preserve the prefix relation among IP addresses. Bloom filter [13] is the only scheme allowing delicate equality test, however, it is also vulnerable to dictionary attacks in the alert correlation case if we create a bloom filter for each IP attribute.

To countermeasure the keyed-hash vulnerability, our intuitive idea is to use *double-random* (i.e., using PEKS to randomize *AlertType* and keyed-hash value of *SrcIPAddress* /

DestIPAddress) in the advanced scheme to provide better privacy protection without losing the capability of equality test of IP attributes. Comparing to the basic scheme, modifications are required to the *setup*, *anonymization* and *alert correlation* procedures. Generally, in the *setup* procedure, extra S-tokens are generated to facilitate the matching of all possible IP addresses, while in the *anonymization* procedure randomness is introduced by PEKS each keyed-hash IP address value as *PEKS(HMAC(*W_{SrcIP}*))* or *PEKS(HMAC(*W_{DestIP}*))*. In the *alert correlation* procedure, besides *AlertType* attribute, the matching algorithm operated on the corresponding *SrcIPAddress* and *DestIPAddress* attributes is also required and works the same as described in Algorithm 1 for *AlertType* attribute to generate HIs.

However, if HMAC is a collision-free hash function, there are totally 2^{32} possibilities of *HMAC(*W_{SrcIP}*)* or *HMAC(*W_{DestIP}*)* for all IPV4 addresses. Thus, in the *setup*, theoretically 2^{32} S-tokens are needed if we output one S-token from each possible IP address so as to eliminate false negative in *alert correlation*. This results in on average 2^{31} matching operations to generate each IP attribute in the generation of HI. Since the matching operation, which includes the computation of bilinear pairing e, is a very time-consuming operation, experienced readers may have noticed that it is not practical to generate HI in this way in reality.

We notice that, the execution times of *matching* operations, T, of each IP address meets following relationship with the S-token space, N (i.e., N is the total number of S-tokens used for matching IP addresses): $T = \left\lceil \frac{32}{\log_2 N} \times \frac{N}{2} \right\rceil$. According to Figure 8, smaller S-token space not only contribute to less storage cost, more importantly, but also lead to fewer matching operations. Thus, we set $N = 2^{x=1} = 2$ to reduce matching operations times to $T = 2^{y=4} = 32$ rather than 2^{31} per IP address. Also note that "x=1" represents in order to achieve S-token space $N = 2$, each IPV4 address should be treated as 32 bits binary.

Thereafter, to anonymize a *SrcIPAddresss* (resp. *DestIPAddresss*) attribute in the *aonymization* phase, the contributor first calculates the *HMAC(*W_{SrcIP}*)* (respectively, *HMAC(*W_{DestIP}*))*

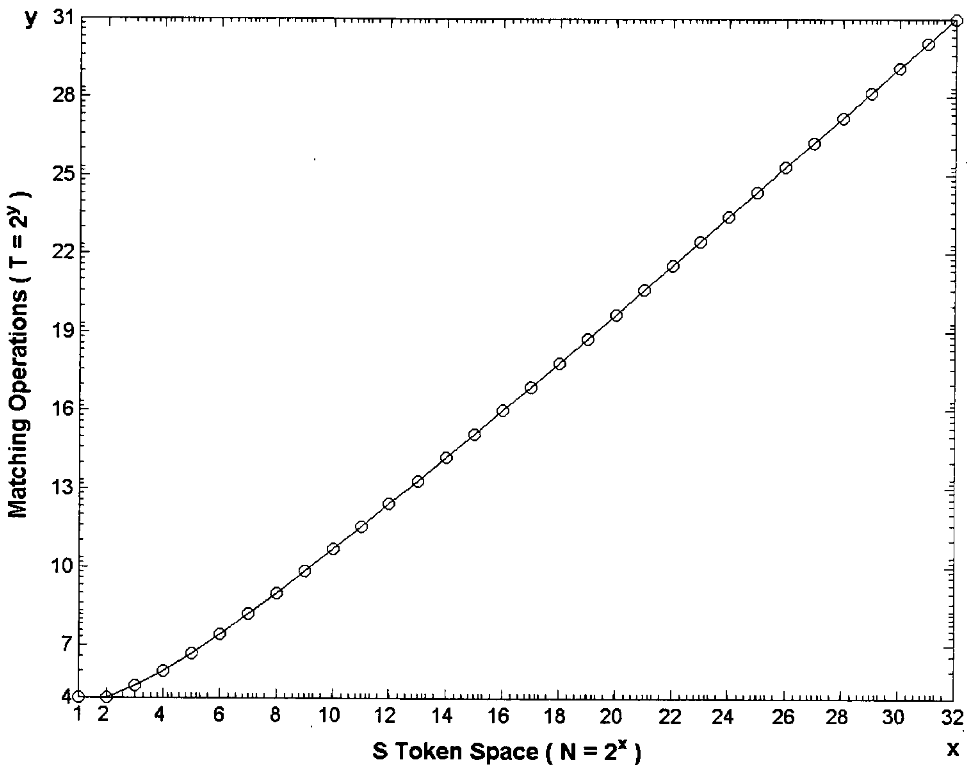


Figure 8: Relationship between S-token Space and Matching Operations

$\in [1, 2^{32}]$. Then transfers the *HMAC*(W_{SrcIP}) (resp, *HMAC*(W_{DestIP})) into 32 bits binary. If *HMAC*(W_{SrcIP})$_p$ (respectively, *HMAC*(W_{DestIP})$_p$) is used to represent the p-th bit of *HMAC*(W_{SrcIP}) (resp, *HMAC*(W_{DestIP})), a anonymized PEKS(*HMAC*(W_{SrcIP})) = PEKS(*HMAC*(W_{SrcIP})$_1$) || PEKS(*HMAC*(W_{SrcIP})$_2$)|| $\cdots$ || PEKS(*HMAC*(W_{SrcIP})$_{32}$) (respectively, PEKS(*HMAC*(W_{DestIP})) = PEKS(*HMAC*(W_{DestIP})$_1$) || PEKS(*HMAC*(W_{DestIP})$_2$)|| $\cdots$ || PEKS(*HMAC*(W_{DestIP})$_{32}$)). The anonymized a will be formated as follows.

$$\boxed{\begin{array}{c} a = I \;||\; \text{PEKS}(W_{rat}) \;||\; \textit{Begin\_time} \;||\; \textit{End\_time} \;||\text{PEKS}(\text{HMAC}(W_{DestIP})) \\ ||\; \textit{DestPort} \;||\; \text{PEKS}(\text{HMAC}(W_{SrcIP})) \;||\; \textit{SrcPort} \end{array}}$$

Accordingly, in the *setup*, S-token generator also generates two trapdoors $T(W_{IP=0})$

Algorithm 2 Privacy-Preserving HIs Generation in the Advanced Scheme

```
Input:
    (1) Anonymized alerts $a_i$, in which
    $\text{PEKS}(W^i_{rat}) = [g^{r_i}, H_2(t_i)]$, and $t_i = e(H_1(W^i_{rat}), h^{r_i})$;
    $\text{PEKS}(\text{HMAC}(W^i_{SrcIP})_p) = [g^{r_i}, H_2(t^s_i)]$, and $t^s_i = e(H_1(\text{HMAC}(W^i_{SrcIP})_p), h^{r_i})$;
    $\text{PEKS}(\text{HMAC}(W^i_{DestIP})_p) = [g^{r_i}, H_2(t^d_i)]$, and $t^d_i = e(H_1(\text{HMAC}(W^i_{DestIP})_p), h^{r_i})$;
    (2) HTs with AlertType $T(W^j_{kat}) = H_1(W^j_{kat})^\alpha$;
    (3) $T(W_{IP=0}) = H_1(W^0_{IP})^\alpha$ and $T(W_{IP=1}) = H_1(W^1_{IP})^\alpha$.
Output: HIs.
1:  Set $i = 1$;
2:  for alert $a_i \in$ Database do
3:      Set $j = 1$;
4:      for $T(W^j_{kat}) \in S$-tokens do
5:          if $H_2(e(T(W^j_{kat}), g^{r_i})) = H_2(t_i)$ then
6:              Set $p = 1$;
7:              for $p \leq 32$ do
8:                  if $H_2(e(T(W^0_{IP}), g^{r_i})) = H_2(t^s_i)$ then
9:                      Replace PEKS(HMAC($W^i_{SrcIP}$)$_p$) by $T(W_{IP=0}), p = p + 1$, goto step 7;
10:                 else
11:                     Replace PEKS(HMAC($W^i_{SrcIP}$)$_p$) by $T(W_{IP=1}), p = p + 1$, goto step 7;
12:                 end if
13:             end for
14:             Set $p = 1$;
15:             for $p \leq 32$ do
16:                 if $H_2(e(T(W^0_{IP}), g^{r_i})) = H_2(t^d_i)$ then
17:                     Replace PEKS(HMAC($W^i_{DestIP}$)$_p$) by $T(W_{IP=0}), p = p + 1$, goto step 15;
18:                 else
19:                     Replace PEKS(HMAC($W^i_{DestIP}$)$_p$) by $T(W_{IP=1}), p = p + 1$, goto step 15;
20:                 end if
21:             end for
22:             Generate HI from HT with $T(W^j_{kat})$, goto step 23;
23:         else
24:             $j = j + 1$;
25:         end if
26:     end for
27:     $i = i + 1$;
28: end for
29: return 0
```

and $T(W_{IP=1})$ that are derived from 0 and 1 bit respectively to match all possible IP addresses besides generating S-tokens from *AlertType*. Consequently, in the *alert correlation*, we propose to execute Algorithm 2 to generate HIs in the advanced scheme. Afterward, the actual correlation of HIs can be efficiently achieved as described in the basic scheme and the same as Ning et al's scheme.

4.4 Security and Privacy-related Properties Achieved in TEIRESIAS

In this section, we first review several attacks specialized against previous works of privacy-preserving information sharing. Then, the investigation of how the security and privacy-related goals defined in Section 3.2 achieved in TEIRESIAS are given.

4.4.1 Attacks against Privacy-preserving Information Sharing

- **Eavesdropping and Casual Browsing** By sending partial anonymized data via public network, and publishing analysis results online, or returning to all (or a set of all) legitimate users may cause released data to be eavesdropped or browsed, possibly even copied, storied and shared out of control capability. Although these kinds of attacks are easy to defend through cryptographic approaches, such as public-key encryption or keyed-hash function we used in TEIRESIAS, potential harms still exist since leaked information may lead more sophisticated attacks which we are going to discuss in the following.

- **Dictionary Attacks** Besides standard hash functions are vulnerable to this kind of attack, a keyed-hash function is also fragile when the secret key used to generate hashes is compromised. The attacker can pre-organize his references by computing all possible values, and then matches with information either eavesdropped from network traffic or published by analysis centers. This kind of attack is especially powerful in some particular situations. For example, if an insider targets a machine in a particular subnet, he can just pre-compute hashes of all possible IP addresses in this network and check through browsed or eavesdropped fragments of network traffic to see if any attribute matches. Since the IP space is usually not enormous for an insider, this kind of attack is very practicable to compromise identity information.

- **Camouflage** If a fully-trusted analysis center exists, an attacker may be highly motivated to set up his own camouflage or compromise an existing one in order to get immediate access to the shared information since it eliminates the need for all other attacks. A similar risk also exists if a center itself is semi-trusted and able to access partially anoymized data.

- **Probe-Response** Probe-response attacks are the nightmare of previous privacy-preserving information sharing schemes, which are based on various anonymization technologies including black marker, truncation, permutation, hash, keyed-hash, bloom filter, prefix-preserving permutation and so on.

  In contrast to most attacks aiming to bypass security guarding devices, these kind of attacks try to be detected and reported. It creates a paradox of privacy-preserving information sharing: for accurate analysis, more detailed local events should be revealed; for strict privacy guarantee, detected local observations should not always be honestly reported. There are two types of probe-response attacks [63] according to the different goals: the first aims to identify the identities of the collaboration monitors [10, 62] while the second aims to learn the security posture of a particular target [46].

4.4.2 Security and Privacy-related Properties Achieved

- **Data Confidentiality** The construction of TEIRESIAS only involves cryptography primitives with well-defined notation of security, and the building blocks (i.e., SEKS and PEKS) have been formally proved secure in [15,68]. Thus, the proposed protocol preserves provable security.

  A curious insider, outsider, or semi-trusted correlator can launch eavesdropping or casual browsing attacks by looking for familiar value of attributes through captured fragments. However, TEIRESIAS is able to fully mitigates this kind of attack since cryptographic primitives in TEIRESIAS guarantee the confidentiality of each anonymized attribute. Moreover, our retrieval mechanism only returns pertinent R-token pairs to avoid disclosuring any information about the original alerts.

  An outsider may also try to launch a correlator camouflage attack by setting up a faked correlator. However, cryptographic primitives assure that this kind of correlator cannot tell any plaintext information by just observing ciphertext. Meanwhile, without receiving transformed knowledgebase from the S-token generator, faked correlator is not able to perform any one of the procedures defined in TEIRESIAS.

- **Unlinkability** In this thesis, we concern two types of unlinkability: identity unlinkability and alert attributes unlinkability.

  1. **Identity Unlinkability** Generally, the possible methods of breaking identity unlinkability can be divided into two categories: traffic-based analysis and protocol-based analysis. The idea behind traffic-based analysis is to detect common information among sniffed network traffic, assuming that any two packets are transferred along the same route if they have information in common. The "common information" could be either identical content (e.g., the same sequence number) in sniffed packets, or identical time consumed in handling

sniffed packets (i.e., time analysis). In TEIRESIAS, to prevent traffic analysis [56], on the one hand, all contributors send their encrypted alerts through a Mixnet-based anonymous communication system such as Tor [23]. On the other hand, each contributor sends (or receives) retrieval requests (or responses) by building a secure channel with the data server so that not only outsiders but also other insiders cannot discover the identity of that contributor.

In protocol-based analysis, adversaries try to deduce the identity of the sender by analyzing the semantic context of messages. For example, to ensure that the receiver can verify the signature, the sender may include her public key in the packet. If that is the case, adversaries can easily discover the identity of the sender. In TEIRESIAS, within any encrypted alert there is no public key and the only identity-related information is R-token, however, it is provable secure and untraceable from the semantic of a R-token itself. Moreover, both the retrieval request and response messages, are sent through secure channels to avoid accessibility of any other insiders or outsiders during the report retrieval procedure.

Even the adversary gets to know all $E(N)$ revealed by a contributor to the data server in retrieval queries, he still can not use any known $E(N)$ to find the x-bits secret of the contributor and thus deduce her identity.

Given that the elliptic curves DLP is hard, TEIRESIAS is also robust against brute force attacks. More specifically, for any PEKS ciphertext given g and g^{α}, there is no PPT function which adversaries can use to find secrets of the contributor (i.e., α) and thus deduce her identity.

2. **Alert Attributes Unlinkability** An insider may analyze the semantic context of messages in order to link alert attributes (i.e., *AlertType*, *SrcIPAddress* and *DestIPAddress*) generated from same plaintext when he eavesdrops some

encrypted alerts from other contributors.

In TEIRESIAS, each attribute encrypted by PEKS is processed with a different secret number α chosen by the contributor. Given that DLP is hard, an insider cannot deduce α from an encrypted attribute, and further to use his encryption ability to launch a dictionary attack so as to link to the original plaintext.

In our basic scheme, IP addresses are encrypted using a uniform keyed-hash function under the same key for all contributors, it is natural vulnerable to an insider and assailable by the correlator or any outsider once secret key is leaked to them. Therefore, in the advance scheme we fix this vulnerability by further using PEKS to process all keyed-hash values to mitigate the possibility of dictionary attacks.

A sophisticated attacker may try to launch probe-response attacks. The first type of probe-response attack greatly relies on analysis centers' public statistical results which should be easily accessed. These public results potentially helps attackers get well organized oracle to identify created probes. An outsider may try to launch this type probe-response attacks. However, the retrieval mechanism we designed can beat these probe-response attacks by excluding any probes included and accessible by the outsider. Note that an insider may not risk himself to launch this type of probe-response attacks since it requires that insider launching active attacks.

The second type of probe-response attacks focus on known IP addresses and it requires launching less probes (i.e., active attacks). However, this kind of delicate attack is still meaningless for an insider since it is highly risky for him to launch active attacks again. Meanwhile, an outsider may try to compromise a correlator in order get access to the probes. To fix this flaw, on the one hand, we design our system as the correlator only can receive incoming network traffic and all outgoing network traffic initiated by the correlator is blocked. On the other hand, we also assume there exist security facilities deployed to protect and monitor the correlator, which is able to be detected and recovered soon once got compromised.

4.5 Resistance to Collusion Attacks

Little attention is paid on the resistance of collusion attacks in the previous works of privacy-preserving information sharing. In the following, we consider the collusion between insiders and outsiders to launch skillful probe-response attacks and give the countermeasure.

To launch probe-response attacks, there are two prerequisites: one is able to launch active attacks as probes and the other is able to access responses from targets so as to identify the artificial probes again. In TEIRESIAS, on the one hand, an outsider is able to launch active attacks and use them as probes. However, our proposed retrieval mechanism can prevent the outsider from viewing his initiated probes again. On the other hand, an insider is able to access the retrieval reports while failing to launch active attacks. So, an insider and an outsider may collude to launch a three-step second type probe-response attack, in which the outsider starts probes and the insider identifies the probes from the retrieved reports.

We use a concrete example to illustrate the attacks. Suppose an attacker, Eve, who is an outsider with IP address *202.077.162.213* would like to inspect whether there is *Sadmind_Amslverify_overflow* vulnerability on a particular machine, Alice, whose IP address is *172.016.112.050*. Bob, is an insider colluding with Eve to launch this attack. We also assume Alice, Eve and Bob all know the HTs described in Example 1.

- First, Eve launches a *Sadmind_Amslverify_overflow* attack directly to the Alice, who generates the first alert in Table 1 to represent Eve's attack.

- Second, Bob creates an artificial alert as the second alert in Table 1. Then Bob acts as a normal contributor anonymizing the generated alert as the second alert in Table 2.

- Third, Bob uses [$E(N_{63997}),\ k_{63997} = f_k(E_1(N_{63997}))$] to query the correlated results about the artificial alert. Ideally, Bob retrieves a following R-token pair:

$$\boxed{E'(C_{N_{63963}}) \rightarrow !||C_{N_{63997}}}$$

  Based on above R-token, Bob can recover the following attack scenario:

$$\boxed{E'(C_{N_{63963}}) \xrightarrow{Prepare-For} a'_{63997}}$$

  Then Bob and Eve are able to work together and conclude there is a good chance that the alert about Eve's attack is correlated with Bob's artificial alert and the real attack scenario is shown as Figure 9. By observing the correlated alerts, Bob and Eve are able to identify that Alice is vulnerable to the *Sadmind_Amslverify_overflow* attack.

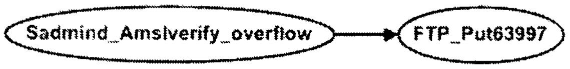


Figure 9: The Attack Scenario Reconstructed by the Collusion Attack

Apparently, the above three-step attack is more skillful and difficult to detect since it just requires the outsider Eve to launch one active attack once, while the insider Bob always operates as a normal contributor. Fortunately, the artificial alert is also correlated with the alert contributed by Alice at the correlator side. Therefore, in the report retrieval procedure, the correlated R-token pair is returned to Alice as well. Consequently, the attack is still able to be detected if Alice studies the retrieval report together with local network activity logs.

More specifically, the above correlated R-token pair is formatted as following to return to Alice:

$$\boxed{!||C_{N_{63963}} \rightarrow E^{'}(C_{N_{63997}})}$$

Based on the R-token, Alice is able to recover the following attack scenario:

$$\boxed{a^{'}_{63963} \xrightarrow{Prepare-For} E^{'}(C_{N_{63997}})}$$

Since the knowledgebase is known by Alice as well, she is able to notice that there is much chance that the encrypted R-token $E^{'}(C_{N_{63997}})$ is related to an *FTP_Put* attack happened on her as shown in Figure 10. Through checking local network activity logs, Alice is able to tell whether it is a really happened attack or a faked alert.

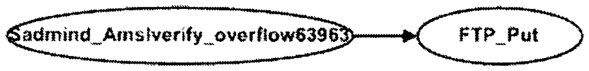


Figure 10: The Attack Scenario Reconstructed to Discover the Collusion Attack

Chapter 5

Implementation and Experiments of TEIRESIAS

The implementation of TEIRESIAS is proceeded in three steps. The first step focuses on the reimplement of a C++ version correlator based on the Ning et al's correlation scheme. The second step involves the construction of function-level applications of SEKS and PEKS written in C++. In the third setup, through integrating function calls to the correlator, which are two blocks developed in the previous two steps, we realize user-level applications of TEIRESIAS. These applications perform operations of different procedures in TEIRESIAS and are provided to the different end-user (i.e., contributor, S-token generator, correlator, and data server).

More specifically, the construction of SEKS application is based on the following cryptographic primitives: pseudorandom generator G and pseudorandom function F are built by using XCBC-MAC [29]; deterministic encryption algorithm $E(N)$ works with AES in ECB mode. The construction of PEKS application leverages the Identity Based Encryption [14] algorithms implemented in the MIRACL library [2]. In the applications, we adopt the well-known Tate Pairing [58], which is the heart of the PEKS encryption and test processes. Based on the MIRACL library, we create an extension that performs PEKS operations, in which a 512-bit prime p is used for effective 1024-bit security, and G_1 and G_2 are groups on the supersingular elliptic curve $y^2 = x^3 + x \bmod p$ with 160-bit group order $q = 2^{159} + 2^{17} + 1$ a prime which divides p. We also use 2^{32} bits randomness, and SHA-256 for hash operations.

All simulations are performed on a desktop with hardware setting Intel(R) Core (TM)2 2.13GHz CPU (64-bit processor) and 2GB RAM. The programs run on Windows XP Professional operation system with ADO database connection to a Microsoft SQL 2000 database server.

5.1 Efficiency Issues

In this section, we describe several approaches that are utilized in our implementation to improve the efficiency of TEIRESIAS.

5.1.1 Bilinear Pairing

The most expensive computational operations used in TEIRESIAS are the calculation of bilinear pairings. In our implementation, the time consumption of a pairing operation $e(P,\ Q)$ (P is a point on the curve over F_p, Q is a point on the quadratic extension field F_{p^2}) is 13 milliseconds.

In TEIRESIAS, the operations of bilinear pairing are utilized in two phases. i.e., to anonymize *AlertType* attribute by calculating $t = e(H_1(W_{rat}, h^r))$ in the anonymization phase, and to match PEKS ciphertexts with S-tokens so as to generate HIs by computing $H_2(e(T(W^j_{kat}),\ g^{r_i}))$ (in Algorithm 1, or $H_2(e(T(W^j_{kat}),\ g^{r_i}))$ and $H_2(e(T(W^0_{IP}),\ g^{r_i}))$ in Algorithm 2) in the alert correlation phase, respectively.

In the anonymization phase, an alternative approach to speed up the encryption is to use pre-computation by storing the fixed pairing parameter (i.e., h^r in $e(H_1(W_{rat}, h^r))$ if a contributor chooses to use a same random secret r for a bunch of alerts). This approach can enhance the computation of a Tate pairing 57% faster than without pre-computation in our implementation.

Nevertheless, the efficiency of the correlation procedures is the key in reality. In both schemes, it takes $T_{Time} = T_{Prerequisite} + T_{Consequence}$ to generate HIs for all incoming alerts, where

$$T_{Prerequisite} = T_{Pothers} + \sum_{i=1}^{N_{PEKS}} T_{Rate} \times N^{i}_{PTokens}$$

$$T_{Consequence} = T_{Cothers} + \sum_{i=1}^{N_{PEKS}} T_{Rate} \times N^{i}_{CTokens}$$

In the above equations, T_{Rate} is defined as the time consumption to perform one time PEKS ciphertext and S-token matching, which includes one time pairing computation; N_{PEKS} is the number of PEKS ciphertext attributes in all anonymized alerts; $N^{i}_{PTokens}$ and $N^{i}_{CTokens}$ are the numbers of S-tokens have been tested before discovering the right match of each PEKS ciphertext to generate the corresponding prerequisite and consequence attribute, respectively. $T_{Pothers}$ and $T_{Cothers}$ are the time consumptions of the rest of the operations (e.g., Hash) to generate prerequisites and consequences, respectively. Note that pre-computation cannot be applied here since the pairing parameters are not fixed. However, we can speed up by deploying two parallel servers to generate the prerequisite and consequence instance sets and the final time consumption is $MAX\{T_{Prerequisite}, T_{Consequence}\}$.

Recently, researchers proposed various optimization approaches to enhance the efficiency of bilinear pairing, such as choosing different groups, computation fields or pairing algorithms, or even, a combination of these approaches. Regarding TEIRESIAS, the current implementations are also limited by our present knowledge and devices: if adopting more efficient pairing algorithms like *Eta* Pairing [38] or hardware implementation of some pairing algorithms like the jobs of Bilinear Pairing IP Core [40] which realized Eta_T pairing over $F_{3^{97}}$ in just 8.17 microseconds (μs) for a pairing computation, better efficiency can be expected from our system. For this reason, we also treat all pairing as "black box" and calculate our system's time consumption without pairing in section 5.3, to further estimate the expected efficiency performance of our system. We leave the job of more detailed optimizations to our further researches and implementations.

5.1.2 Indexing

The *indexing* technique adopted in our implementation is able to benefit TEIRESIAS from two aspects: achieving better efficiency (by minimizing bilinear pairing times) and providing stronger privacy protection (of knowledgebase).

The idea is based on the observation that the S-token list consists of two types of tokens as shown in Table 3. Type A tokens are generated from all possible *AlertType*, and type B tokens are the possible prerequisites or consequences of each *AlertType*. More specifically, any *AlertType* attribute generated by an IDS belongs to type A, while a token belonging to type B is used to link two type A alerts in the correlation algorithm. Be aware that, at the S-token generation phase, we index all S-tokens accordingly. Afterward, we use the indexes of type B tokens to indicate prerequisites and consequences of a type A token instead of ciphertexts.

Therefore, when generates a Hyper-alert instance, once an alert's *AlertType* has been matched with a type A S-token, instead of spending extra matching operations several more times to identify its' prerequisite and consequence attributes by matching with type B tokens, we just need to copy the index numbers indicated in the knowledgebase and put them in the corresponding attributes, which greatly decrease the computation workload caused by matching operations.

Meanwhile, indexing of prerequisites and consequences can provide better knowledgebase privacy protection. Since in the knowledgebase and any one of HIs, the representation of prerequisite and consequence attributes are just randomly assigned index numbers by the S-token generator without patterns to be tested by attackers, which means no one is able to learn the knowledgebase masked by the S-token generator simply by observing the indexed knowledgebase.

Table 3: Classified and Indexed S-Tokens

Index No.	S-token	Type
...	...	...
5	$T(ExistService)$	B
6	$T(GainAccess)$	B
15	$T(OSSolaris)$	B
22	$T(SystemCompromised)$	B
25	$T(VulnerableSadmind)$	B
33	$T(FTP\_Put)$	A
45	$T(Sadmind\_Amslverify\_overflow)$	A
...	...	...

5.1.3 Example 3

With index numbers, HTs in example 2 can be represented as following:

$$T(Sadmind\_Amslverfiy\_overflow) =$$
$$(\{DestIP\},$$
$$25(DestIP) \wedge 15(DestIP),$$
$$\{6(DestIP)\})$$

$$T(FTP\_Put) =$$
$$(\{DestIP, DestPort\},$$
$$5(DestIP, DestPort) \wedge 6(DestIP),$$
$$\{22(DestIP)\})$$

The above HTs facilitate correlator generating following HIs:

$$PEKS(Sadmind\_Amslverfiy\_overflow) =$$
$$(\{HMAC(172.016.112.050)\},$$
$$25(HMAC(172.016.112.050)) \wedge 15(HMAC(172.016.112.050)),$$
$$\underline{\{6(HMAC(172.016.112.050))\}}).$$

$$PEKS(FTP\_Put) =$$
$$(\{HMAC(172.016.112.050), 32773\},$$
$$5(HMAC(172.016.112.050, 32773)) \wedge \underline{6(HMAC(172.016.112.050))},$$
$$\{22(HMAC(172.016.112.050))\})$$

Similarly, the underlined parts of above HIs guide correlating these two alerts. It is also can be demonstrated from the above expressions, on the one hand, the bilinear pairing times are reduced by directly using an index number to represent each type B S-token. On the other hand, the accuracy is preserved and privacy is enhanced during correlation procedure.

5.2 Experiments

To evaluate the efficiency and accuracy of adopting the TEIRESIAS protocol in correlating IDS alerts, we perform a series of experiments using the DARPA 2000 intrusion detection scenario specific datasets (LLDOS 1.0 and LLDOS 2.0.2). LLDOS 1.0 contains a series of attacks in which an attacker probes, breaks-in, and installs the components necessary to launch a Distributed Denial of Service (DDOS) attack, and actually launches a DDOS attack against an off-site server. LLDOS 2.0.2 includes a similar sequence of attacks run by an attacker who is more sophisticated than the trusted one.

Table 4: Correlation Results of Ning et.al's Scheme

	LLDOS 1.0			**LLDOS 2.0.2**		
Dataset(Alerts No.)	DMZ (891)	Inside (922)	DMZ+Inside (1813)	DMZ (430)	Inside (494)	DMZ+Inside (924)
$\#_{Alerts}$	57	44	102	5	13	19
$\#_{Pairs}$	125	161	446	3	29	37
T_{Time} (second)	3	3	7	2	2	3

As recorded in Table 4, Table 5 and Table 6, we conduct the following three sets of experiments based on the alerts generated by a RealSecure Network Sensor [39] which is feed with DARPA 2000 datasets. First, to set the reference, we test the Ning et.al's correlation scheme in our environment (in Table 4). Second, we process alerts by using the basic scheme (in Table 5 and Table 6). Third, considering the particular circumstances when the IP addresses in alerts can be abstracted into a same external network address, we can choose only PEKS part of the binaries from the HMAC of the original IP address to further improve the efficiency of HIs generation. In the advanced scheme 1 and advanced scheme 2 (in Table 5 and Table 6), we set to PEKS all 32 bits and last 8 bits IP binaries, respectively.

In Table 4, we record $\#_{Alerts}$ as the total number of alerts extracted from alert sets; $\#_{Pairs}$ denotates number of the alert couples discovered from extracted alerts; $T_{Time} = MAX\{T_{Prerequisite}, T_{Consequence}\}$ represents the time needed to generate HIs. In Table 5 and Table 6, we also record $\#'_{Alerts}$, $\#'_{Pairs}$ and T'_{Time} to represent results w.r.t correlation algorithm of basic scheme and advanced schemes on anonymized alerts, respectively. Without loss of consistency, we continue use symbols with the superscript ($'$) to denote results w.r.t TEIRESIAS following. $\#'_{Test} = \{\#'_{Prerequisite}, \#'_{Consequence}\}$ are the total number of test functions called to match all the PEKS attributes (i.e., *AlertType*, *SrcIPaddress* and *DestIPaddress*), while $\#'_{AlertType} = \{\#'_{Prerequisite}, \#'_{Consequence}\}$ are the number of test functions called to match *AlertType*. Similarily, $T'_{Test} = MAX\{T'_{Prerequisite}, T'_{Consequence}\}$ are the total time consumption of the all matching functions and $T'_{Pairing} = MAX\{T'_{Prerequisite}, T'_{Consequence}\}$ are the total time consumption of bilinear pairings which are included in the matching functions.

Table 5: Correlation Results of TEIRESIAS and Usability Metrics

Scheme	DataSet (Alert No.)		$\#'_{Alerts}$	$\#'_{Pairs}$	$\#'_{Test}$	$\#'_{Test(AlertType)}$	M_{CC}^{Marco}	M_{CC}^{Micro}
Basic Scheme	**LLDOS 1.0**	DMZ (891)	57	125	{ 7165, 8099 }	{ 7165, 8099 }	100%	125
		Inside (922)	44	161	{ 7266, 8561 }	{ 7266, 8561 }	100%	161
		DMZ+Inside (1813)	102	446	{14431, 16660}	{14431, 16660}	100%	446
	LLDOS 2.0.2	DMZ (430)	5	3	{ 3126, 3618 }	{ 3126, 3618 }	100%	3
		Inside (494)	13	29	{ 3881, 4560 }	{ 3881, 4560 }	100%	29
		DMZ+Inside (924)	19	37	{ 7007, 8178 }	{ 7007, 8178 }	100%	37
Advanced Scheme 1	**LLDOS 1.0**	DMZ (891)	57	125	{ 34269, 49667 }	{ 7165, 8099 }	100%	125
		Inside (922)	44	161	{ 36706, 51249 }	{ 7266, 8561 }	100%	161
		DMZ+Inside (1813)	102	446	{ 70975, 100916}	{14431, 16660}	100%	446
	LLDOS 2.0.2	DMZ (430)	5	3	{ 16566, 23906 }	{ 3126, 3618 }	100%	3
		Inside (494)	13	29	{ 19401, 26928 }	{ 3881, 4560 }	100%	29
		DMZ+Inside (924)	19	37	{ 35967, 50834 }	{ 7007, 8178 }	100%	37
Advanced Scheme 2	**LLDOS 1.0**	DMZ (891)	57	125	{13941, 18491}	{7165, 8099}	100%	125
		Inside (922)	44	161	{14626, 19233}	{7266, 8561}	100%	161
		DMZ+Inside (1813)	102	446	{28567, 37724}	{14431, 16660}	100%	446
	LLDOS 2.0.2	DMZ (430)	5	3	{6486, 8690}	{3126, 3618}	100%	3
		Inside (494)	13	29	{7761, 10152}	{3881, 4560}	100%	29
		DMZ+Inside (924)	19	37	{14247, 18842}	{7007, 8178}	100%	37

5.3 Evaluation Metrics

We use four metrics to evaluate alert correlation with TEIRESIAS in comparison with Ning et.al's scheme.

Table 6: Time Consumptions and Efficiency Metrics

Scheme	DataSet (Alert No.)		T'_{Time} (second)	T'_{Test} (second)	$T'_{Pairing}$ (second)	$T'_{Expected}$ (second)	$M_E \leq$	$M_{EE} \leq$
Basic Scheme	**LLDOS 1.0**	DMZ (891)	112	108	107	5	37	2
		Inside (922)	118	114	112	6	39	2
		DMZ+Inside (1813)	230	222	219	11	33	2
	LLDOS 2.0.2	DMZ (430)	50	48	47	3	25	2
		Inside (494)	63	61	60	3	32	2
		DMZ+Inside (924)	113	109	107	6	38	2
Advanced Scheme 1	**LLDOS 1.0**	DMZ (891)	693	669	659	35	231	11
		Inside (922)	715	690	681	35	238	11
		DMZ+Inside (1813)	1416	1358	1338	79	202	11
	LLDOS 2.0.2	DMZ (430)	333	322	317	16	167	8
		Inside (494)	375	362	357	18	188	9
		DMZ+Inside (924)	709	683	673	37	236	12
Advanced Scheme 2	**LLDOS 1.0**	DMZ (891)	261	247	244	14	87	5
		Inside (922)	272	257	253	19	91	6
		DMZ+Inside (1813)	538	506	498	40	77	6
	LLDOS 2.0.2	DMZ (430)	123	116	114	9	62	5
		Inside (494)	143	136	134	9	72	5
		DMZ+Inside (924)	266	253	249	17	89	6

- Metrics M_{CC}^{macro} and M_{CC}^{micro}: **Usability**

  we use metric M_{CC}^{macro} and metric M_{CC}^{micro} from both macro and micro aspects to evaluate the *Completeness* (All valid correlation alerts discovered by TEIRESIAS are the same as Ning et.al's scheme) and *Correctness* (Each correlated alert pair discovered by the TEIRESIAS is a valid link and is the same as the discovery of Ning et.al's scheme).

  $M_{CC}^{macro} = \frac{\#_{Alerts}}{\#'_{Alerts}} \times \frac{\#_{Pairs}}{\#'_{Pairs}}$ is defined as a percentage which involves number of extracted alerts and number of coupled alerts.

  $M_{CC}^{micro} = \sum_{i=1}^{\#'_{pairs}} M_i$ is defined to dig into each discovered prepare-for alert pair and M_{CC}^{micro} equals to the number of exactly same matched couples in Teiresias compared to Ning et.al's scheme. For any (one of $\#'_{pairs}$) HI pair $P'_i \longrightarrow C'_i$ correlated by TEIRESIAS, if we can find a HI pair $P_i \longrightarrow C_i$ correlated by Ning et.al's scheme have the same global *AlertID* assigned during per-process process, the corresponding $M_i = 1$, otherwise $M_i = 0$.

- Metric M_E: **Efficiency**

$M_E = \frac{T'_{Time}}{T_{Time}}$ evaluates the *Efficiency* through time consumption in TEIRESIAS compared to that of Ning et al's scheme.

- Metric M_{EE}: **Expected Efficiency**

  By adopting various technologies we can achieve more efficient performance to power the system. Here, we estimates the *Expected Efficiency* $M_{EE} = \frac{T'_{Expected}}{T_{Time}}$ by adopting hardware implementation of bilinear pairing as the work in [40] instead of our current software implementation. $T'_{Expected}$ can be computed as $T'_{Time} - T'_{Pairing} + \#'_{Test} * (8.17 \times 10^{-6})$.

5.4 Analysis of Experimental Results

Through usability metrics, in Table 4 and Table 5, $M_{CC}^{Macro} = 100\%$ and $M_{CC}^{Micro} = \#_{Pairs} = \#'_{Pairs}$ for each dataset each scheme demonstrates our system with TEIRESIAS still preserves no wore usability than the origional correlation algorithm based on these datasets.

Through the efficiency metrics, in Table 6, the comparison of M_E and M_{EE} illustrates that the efficiency was enhanced by adopting more more efficient pairing implementations. More specifically, to handle around 1000 alerts without pairing will cost $T'_{Time} - T'_{Pairing} \approx 6, 36$ and 17 seconds for three schemes, respectively. Based on expected hardware environment, we can save around $95\%, 94\%$, and 93% time for three schemes compared to the software implementation. i.e., around 1000 alerts cost no more than 6 seconds, 37 seconds and 17 seconds for the three schemes, respectively. Moreover, we find that test functions cost majority of total time through evaluating $\frac{T'_{Test}}{T'_{Time}} \geq 96\%, 96\%$ and 94% for three schemes, respectively. The fact that most of the time computations are pairings is further proven by evaluating $\frac{T'_{Pairing}}{T'_{Test}} \geq 98\%, 98\%$ and 98% for all three schemes, respectively.

Two more efficiency related observations from experiential results are:

- In Table 5, $\#'_{Test} = \#'_{Test(AlerType)}$ for the basic scheme since all matching computations are used to math *AlertType* attribute. We notice that the average pairing times which can be computed by $\frac{\#'_{Test.Prerequisite} + \#'_{Test.Consequence}}{Alert\ No.}$ for advanced schemes 1 and 2 are roughly increased to 4 and 1 times compared to the basic scheme. The underlying reason is that the average matching times of the basic scheme is about 16 (including generating prerequisite and consequence instance), which equals to the pairing times for each additional 8 bits of both $SrcIPAddress$ and $DestIPAddress$ binary.

- In Table 5, the increase of matching times, which equals to $\#'_{Test}$, is not a strict linear function that has alert numbers as input variable. The reason is two-fold: first, the knowledgebase is not (the knowledgebase in our simulation is defined for research purposes) and it is not possible (new attacks emerge frequently) to defined comprehensively. The consequence is that once an *AlertType* attribute cannot be recognized, the matching operations of IP address parts can be ignored. Actually, 60 out of 891 alerts from DMZ dataset and 24 out of 922 alerts from Inside dataset can not be identified in LLDOS 1.0 scenario; 11 out of 430 alerts from DMZ dataset and 12 out of 494 alerts from Inside dataset cannot be identified in LLDOS 2.0.2 scenario. Second, the knowledgebase itself affects pairing times. For example, to generate a Hyper-alert Instance after identifying knowledgebase *T(Mstream_Zombie)=({DestIP, SrcIP}, 22(DestIP)* $\wedge$ *22(SrcIP), {17(null)})* requires 32 times of pairing computation to identify *SrcIPAddress* and *DestIPAddress*. However, to generate a Hyper-alert Instance after identifying knowledgebase *T(Sadmind_Amslverfiy_overflow) = ({DestIP}, 25(DestIP)* $\wedge$ *15(DestIP), {6(DestIP)})* only requires 16 times of pairing computation to identify *DestIPAddress*. These above two factors can potentially explain the reason why $\#'_{Test_{.Prerequisite}}$ of LLDOS 1.0 DMZ dataset with 891 alerts is even more than $\#'_{Test}$ of LLDOS 2.0.2 DMZ+Inside datasets with 924 alerts.

5.5 Illustration of the Cross-domain Alert Correlation

The charming of cross-domain alert correlation lies that it potentially fuses distributed information together and explores the limited scope of local analysis. In the simulation, we input both DMZ and Inside datasets and the advantage of cross-domain alert correlation can be demonstrated from correlated results in Figure 11. Compared to the correlation results generated from separate DMZ and Inside dataset in Figure 12 and Figure 13, more information has been extracted, which can be reflected by the number of correlated alerts and the number of prepare-for pairs recorded in Table 5.

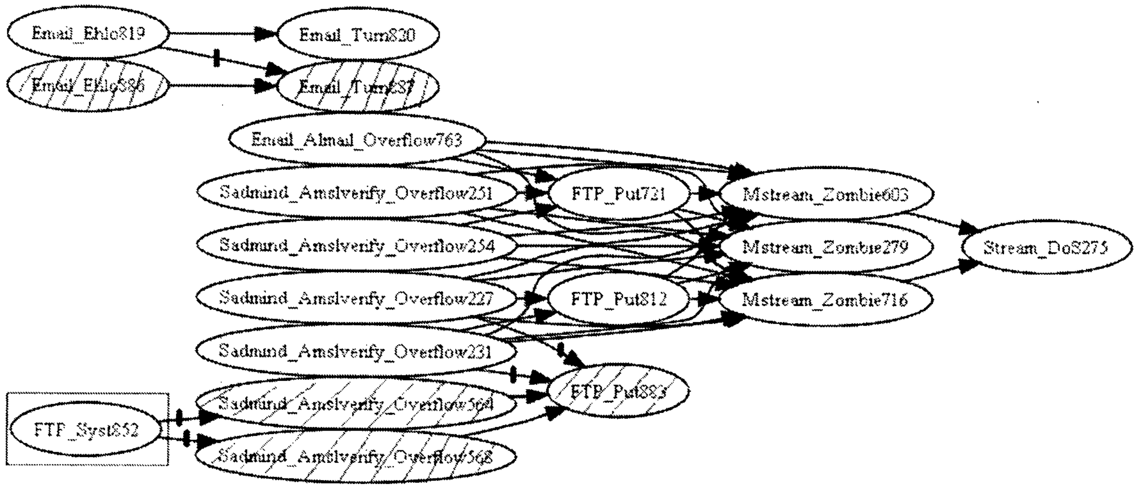
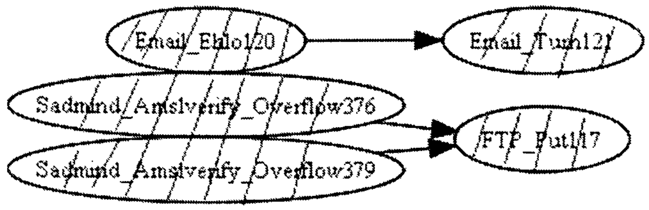


Figure 11: LLDOS 2.0.2 Cross-domain (DMZ+ Inside) Correlation Result

In Figure 11, the ellipses with oblique lines are the alerts from DMZ dataset, and the blank ellipses are the alerts from Inside dataset. The arrows are used to represent prepare-for relationships read from left to right. As reflected, on the one hand, one more alert *FTP_Syst852* is extracted from Inside dataset and correlated with two alerts *Sadmind_Amslverfiy_overflow564* and *Sadmind_Amslverfiy_overflow568* extracted from DMZ dataset,



Email_Ehlo120
Email_Turn121
Sadmind_Amslverify_Overflow376
Sadmind_Amslverify_Overflow379
FTP_Put117

Figure 12: LLDOS 2.0.2 DMZ Dataset Correlation Result

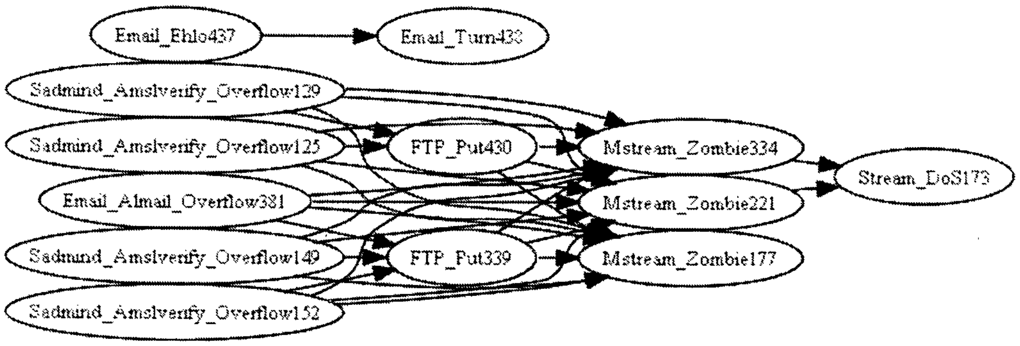


Figure 13: LLDOS 2.0.2 Inside Dataset Correlation Result

on the other hand, five more cross-domain prepare-for couples: *Email_Ehlo819* → *Email_-Turn887*, *FTP_Syst852* → *Sadmind_Amslverfiy_overflow564*, *FTP_Syst852* → *Sadmind_-Amslverfiy_overflow564*, *FTP_Syst852* → *Sadmind_Amslverfiy_overflow568*, *Sadmind_-Amslverfiy_overflow227* → *FTP_Put883*, and *Sadmind_Amslverfiy_overflow23* → *FTP_-Put883* are discovered from extracted alerts, therefore, Figure 11 is not a simply the addition of two subgraphs (i.e., Figure 12 and Figure 13) output by two datasets. A similar observation also can be found by inputing both DMZ LLDOS 1.0 and Inside LLDOS 1.0 datasets as indicating in Appendix A, B and C.

Chapter 6

PEKSrand

In this chapter, we describe and analyze the developed extensions (i.e., PEKSrand-BG and PEKSrand-SG) that provide stronger privacy (i.e., predicate privacy and statistics privacy) protection in PEKS.

6.1 Basic Ideas of PEKSrand

We observe that, PEKS's incompetence in ensuring predicate privacy is due to two facts. On the one hand, the keywords used to create trapdoors in the targeted applications (e.g., intelligent email routing) are meaningful dictionary words. On the other hand, there exists a deterministic one-to-one mapping between a keyword and a trapdoor.

Based on this observation, our first idea is to randomize the original keywords so that the transformed keywords used to generate the trapdoors are not meaningful dictionary words any more. A naïve solution is that, user R (i.e., the receiver) and all other users (i.e., senders) share a secret N, which is concatenated with original keywords (e.g., the key refreshing solution proposed by Baek, Safiavi-Naini, and Susilo [8]) or is used as the key for hashing original keywords. However, such privacy protection is frail, since the protection of the shared secret is difficult. If any sender is compromised, which is very likely given that the size of the set of senders is usually large, this protection relies entirely on the security of the semi-trusted delegate. Moreover, it is also not suitable for scenarios where the membership of the set of senders might be dynamic, which results in additional costs of key/secret management. To address this issue, we limit the entities that hold the secret used for randomization to only one or a small set of proxy servers, which are well protected and thus are more secure than normal senders. This method also greatly reduces the cost of key/secret management. The PEKSrand-BG Scheme is built upon the first idea.

Another idea is to map a keyword to multiple trapdoors instead of one. It can weaken the effectiveness of statistical guessing attacks at the cost of the increasing overhead of storing trapdoors at the semi-trusted delegate. The PEKSrand-SG Scheme is developed through a combination of both ideas.

In our design, besides the three types of entities in the original PEKS system, we add a new type of entities called *proxy server*. In the remained parts of this paper, to avoid confusions, we denote searching server and proxy server as *gateway* and *proxy*, respectively.

We assume that, both the proxy and the gateway are semi-trusted. In other words, they do not launch active attacks (e.g., probe-response attacks [10,46,62]) or collude with any malicious user, unless being compromised. We also assume that there exist certain security mechanisms that can detect the compromise that occurs on any proxy or the gateway and recover it within a short period. Hence, we assume that, although the adversary is capable of compromising the gateway or a proxy, she cannot control both all proxies and the gateway at the same time. We argue that this assumption is reasonable in practice, in particular in the PEKSrand-SG scheme, where a set of proxies instead of one are used.

6.2 The PEKSrand-BG Scheme

The framework of the PEKSrand-BG scheme is illustrated in Figure 14. We denote the receiver and the sender as Alice and Bob, respectively. Now, Bob wants to send a message

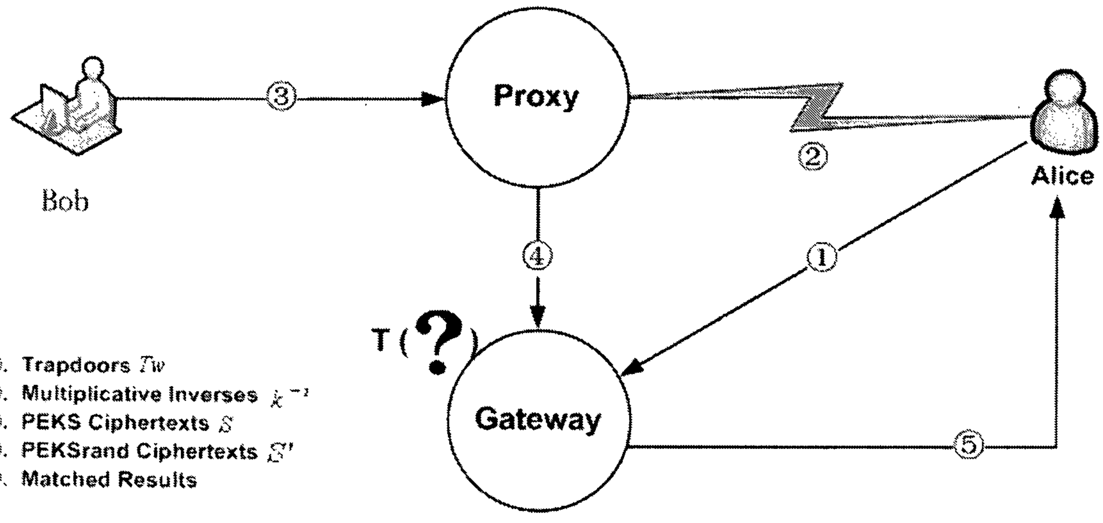


Figure 14: The Framework of the PEKSrand-BG Scheme

(e.g., an email) to Alice, who relies on the gateway to route the incoming messages based on the keywords contained in the messages.

In PEKSrand-BG, to be resistant to brute-force guessing attacks, Alice transforms the original meaningful keywords using a secret during the trapdoor generation. To guarantee that the searching function is still workable with randomized keywords without any interaction between Bob and Alice, we employ a proxy which sits between senders and the gateway. The proxy's major responsibility is to pre-process the PEKS ciphertexts received from the senders before forwarding them. We specify two hash functions $H_1 : \{0,\ 1\}^* \rightarrow G_1$ and $H_2 : G_2 \rightarrow \{0,\ 1\}^{\log p}$. The detailed procedures are as follows.

• $KeyGen(s)$: Alice picks a random number $\alpha \in Z_p^*$ and a generator g of G_1, and then outputs a public/private key pair $A_{pub} = [g,\ h = g^{\alpha}]$ and $A_{priv} = \alpha$. Afterward, Alice chooses a secret number $k \in \mathbb{Z}_p$ and calculates it's multiplicative inverse as $k^{-1} \in \mathbb{Z}_p$ which satisfies $k * k^{-1}(mod\, p) = 1$. At the end of this step, Alice sends k^{-1} to the proxy through a secure channel between Alice and the proxy;

• $Trapdoor(A_{priv}, x, k)$: Given the private key $A_{priv} = \alpha$, the secret k and a keyword x, Alice produces the trapdoor $T_x = H_1(x)^{\alpha * k}$ and delivers it to the gateway through another secure channel between Alice and the gateway;

• $PEKS(A_{pub}, x)$: For a keyword x, Bob first picks a random number $r \in Z_p^*$, and computes $t = e(H_1(x), h^r) \in G_2$, then outputs the PEKS ciphertext $S = [g^r, H_2(t)]$. Then, the PEKS ciphertext S is sent to the proxy;

• $PEKSrand(S, k^{-1})$: For each PEKS ciphertext S received, the proxy updates it with the multiplicative inverse number k^{-1}. More specifically, the transformed PEKS ciphertext (i.e., the PEKSrand ciphertext) is calculated as $S' = [g^{r*k^{-1}}, H_2(t)]$. Afterwards, the proxy forwards S' to the gateway.

• $Test(S', T_x)$: Let each PEKSrand ciphertext $S' = [A, B]$. The gateway tests if $H_2(e(T_x, A)) = B$. If so, then it is a match otherwise it is not match.

6.3 The PEKSrand-SG Scheme

Although PEKSrand-BG is efficient and can defend brute-force guessing attacks, we still have a few concerns about the security of this scheme. In PEKSrand-BG, we raise the threshold of breaking the system through compromising online server(s) from a single gateway in PEKS to two servers (i.e., a gateway and a proxy). However, in security-critical scenarios, we may want to further raise the bar. The other concern is that, the PEKSrand-BG scheme is still vulnerable to statistical guessing attacks. It is due to the fact that, the PEKSrand-BG scheme breaks only the deterministic and direct mapping between a meaningful keyword and the corresponding trapdoor through randomizing the original keyword but not the indirect one-to-one mapping between the original keyword and the new trapdoor. Hence, the frequency of the appearance of a specific keyword is the same as that of the corresponding trapdoor or predicate. Consequently, in the scenarios where the adversary has extra knowledge on the statistical distribution of keywords, the PEKSrand-BG scheme fails to protect predicate privacy.

For the first concern, a naïve solution of maintaining a few proxies holding the same secret k does not work. Even worse, it actually increases the risk of server compromises. Therefore, we think about increasing both the number of proxies and the number of secrets stored among the set of proxies. As to the second concern, our solution is to transform the one-to-one mapping, either direct or indirect, between an original keyword and a corresponding trapdoor in PEKS into a one-to-many mapping. To address these two concerns, in PEKSrand-SG we employ a combination of two methods: *Proxy Farm* and *Random Walk*.

A proxy farm consists of N proxies, each of which stores a distinct multiplicative inverse. In a simple application of this proxy farm method, upon receiving a PEKS ciphertext, the proxy performs the same type of ciphertext transformation as in PEKSrand-BG, with its own multiplicative inverse. Afterwards, the proxy forwards the resulting PEKSrand ciphertext to the gateway. In such a scheme, the PEKSrand ciphertexts corresponding to the same keyword are verified by distinct trapdoors at the gateway, if they are generated by different proxies. In other words, the original one-to-one mapping has been converted into a one-to-F mapping, where F is an important parameter related to privacy protection. As a result, the gateway has to store all the F trapdoors corresponding to the same keyword. Hence, the storage overhead at the gateway is increased by a factor of F. In addition, we need F proxies in the proxy farm. Although the storage overhead is reasonable in practice[1], it is costly to maintain a proxy farm with a large size, considering the level of security protection and trust level required. To mitigate this overhead, we integrate the idea of random walk into the proxy farm method. Now, a ciphertext will be transformed multiple times with distinct inverses instead of only once before it is finally forwarded to the gateway. Let U and u denote the number of proxies in the proxy farm and the number of times that a ciphertext is transformed with distinct inverses. In such a new method, with a proxy form with size U, we can achieve the same level of privacy protection as that is provided by a proxy form

[1]Please refer to Section 6.5.2 for more details.

with size C_U^u in the simple application of this proxy farm method. The framework of the PEKSrand-SG scheme, which incorporates the ideas of proxy forma and random walk, is shown in Figure 15.

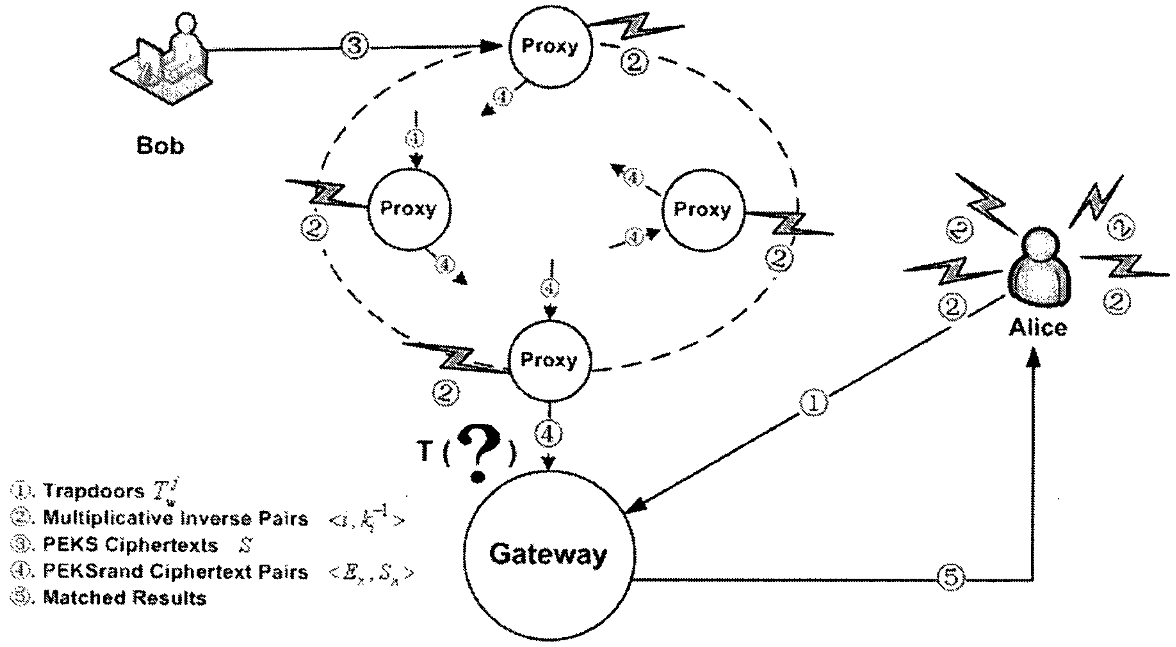


Figure 15: The Framework of the PEKSrand-SG Scheme

6.3.1 Procedures of the PEKSrand-SG Scheme

The PEKSrand-SG scheme consists of the four phases: setup, encrypt, random-walk, and keyword-searching.

- **Setup** To initialize the whole system, the following system-wide parameters are defined: U is the number of proxies that form the proxy farm, while u is the number of distinct proxies involved in a random walk; the security parameter s determines the size, p, of the groups G_1 and G_2, and e is a symmetric bilinear pairing between two groups and defined as $e : G_1 \times G_1 \rightarrow G_2$. Similar to PEKSrand-BG, to generate a system wide public key pair, Alice picks a random $\alpha \in Z_p^*$ and a generator g of G_1, and outputs $A_{pub} = [g, h = g^{\alpha}]$ and $A_{priv} = \alpha$.

To initialize the PEKSrand function in a proxy farm consisting of U proxies, Alice chooses U secret numbers ($k_i,\ for\ i = 1,\ 2, \ldots,\ U$) and calculates the corresponding multiplicative inverses k_i^{-1} that satisfies $k_i * k_i^{-1} (mod\, p) = 1$. Then, Alice sends each proxy a distinct $< i,\ k_i^{-1} >$ pair through a secure channel.

For each keyword x specified by Alice, C_U^u trapdoors corresponding to x, denoted as $T_x^j (j \in \{1, 2, \ldots, C_U^u\})$, are generated. The trapdoors T_x^j's are calculated as follows:

$$T_x^j = H_1(x)^{\alpha * \prod_{i \in V_j} k_i}, \quad for\ j \in \{1, 2, \ldots, C_U^u\} \tag{1}$$

where V_j is a subset of $\{1,\ 2, \ldots,\ U\}$ with u elements. Let I_j denote a string that concatenates all elements in V_j with a predefined delimiter, such as ":". For example, given that $V_j = \{2, 4, 7\}$, I_j is denoted as "2:4:7". Finally, Alice distributes all C_U^u pairs of $< I_j,\ T_x^j >$ to the gateway through a secure channel.

- **Encrypt** In the encrypt phase of PEKSrand-SG, Bob encrypts the keyword x in the same way as in PEKSrand-BG and outputs the PEKS ciphertext $S = [g^r, H_2(t)]$. Afterwards, S is forwarded to a randomly chosen proxy in the proxy farm.

- **Random-Walk** Without loss of generality, we assume that proxy P_1 is the first proxy receiving the PEKS ciphertext S and P_1 holds the inverse k_1^{-1}. P_1 transforms the ciphertext with k_1^{-1} and outputs $S_1 = [g^{r * k_1^{-1}}, H_2(t)]$. Then, proxy P_1 generates a $< E_1,\ S_1 >$ pair, where E_1 is the index of the multiplicative inverse held by proxy P_1 in the format of a string (i.e., "1" in this case), and forwards the pair to a randomly chosen proxy in the farm other than itself.

  Without loss of generality, we assume that the path of the random walk within the proxy farm is $P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_u$, and proxy P_i holds a multiplicative inverse k_i^{-1} for $i = 1, 2, \ldots, u$. For the following random walk process, we denote the PEKSrand ciphertext pair that a proxy P_i receives from another proxy as $< E_x, S_x = [g^{r*k_1^{-1}*k_2^{-1} \cdots *k_x^{-1}}, H_2(t)] >$ where x represents the number of proxies that have performed a transformation on the ciphertext so far during the random walk. Proxy P_i first checks whether the index of its multiplicative inverse is indicated in E_x. If it is true, it means that proxy P_i has previously performed a transformation on this ciphertext. In such a case, P_i simply forwards the received pair to a randomly chosen proxy again without any modification. Otherwise, proxy P_i will update the pair as $< E_{x+1}, S_{x+1} = [g^{r*k_1^{-1}*k_2^{-1} \cdots *k_x^{-1}*k_{(x+1)}^{-1}}, H_2(t)] >$, where E_{x+1} is the concatenation of E_x and the index of the multiplicative inverse of P_i, separated by the predefined delimiter. Afterwards, proxy P_i checks the number of the indexes of inverses that appear in E_{x+1}. If it is less than u, P_i forwards the pair to a randomly chosen proxy in the farm other than itself. If it is equal to u, the random walk process is complete, and proxy P_i will forward the $< E_u, S_u >$ pair to the gateway.

- **Keyword-Searching** The whole trapdoor set that Alice assigns to the gateway can be divided into C_U^u subsets, each of which contains d trapdoors for d keywords that Alice chooses and is corresponding to a unique combination of u proxies. Each subset can be labeled with the corresponding I_j for $j \in \{1, 2, \ldots, u\}$. Upon receiving a $< E_u, S_u >$ pair from the last hop of the proxy farm, instead of searching the whole trapdoor set, therefore, we may first identify the subset of trapdoors corresponding to the combination of proxies that have performed the transformation operation on the ciphertext. It can be done by simply comparing X_u with the I_j's of subsets that the gateway receives from Alice.

  Once the subset of trapdoors is determined, the gateway performs the keyword searching step in the same way as in the PEKSrand-BG scheme. Let $S_u = [A, B]$ denote the received PEKSrand ciphertext. More detailedly, the gateway executes $Test(S_u, T_x^j)$ to verify whether $H_2(e(T_x^j, A)) = B$ is satisfied. If so, it means that the original plaintext contains the keyword corresponding to the trapdoor used in the verification, i.e., x. The correctness of the verification is shown as follows.

$$\begin{aligned}
H_2(e(T_x^j, A)) &= H_2(e(H_1(x)^{\alpha*k_1*k_2*\cdots*k_n}, g^{r*k_1^{-1}*k_2^{-1}*\cdots*k_n^{-1}})) \\
&= H_2(e(H_1(x), g)^{\alpha*k_1*k_2*\cdots*k_n*r*k_1^{-1}*k_2^{-1}*\cdots*k_n^{-1}}) \\
&= H_2(e(H_1(x), g)^{\alpha*r*k_1*k_1^{-1}*k_2*k_2^{-1}*\cdots**k_n*k_n^{-1}}) \\
&= H_2(e(H_1(x), g)^{\alpha*r}) \\
&= B
\end{aligned}$$

6.4 Security and Privacy Analysis of the PEKSrand Schemes

In this section, we analyze the level of security and privacy achieved in the PEKSrand-BG and PEKSrand-SG schemes.

6.4.1 Security Analysis

The security of both PEKSrand schemes relies on the difficulty of the Elliptic Curve DLP: suppose g^x and $g^{x*k^{-1}}$ (resp. g^{x*k}) are two points on an elliptic curve where both k^{-1} (respectively, k) and x are scalars. Given g^x and $g^{x*k^{-1}}$ (resp. g^{x*k}), it is computationally infeasible to obtain k^{-1} (respectively, k) , if k^{-1} (resp. k) is sufficiently large.

In the PEKSrand-BG scheme, due to the usage of randomized keywords, in order to break the system, e.g., compromising data confidentiality, the adversary has to compromise both the gateway and the proxy. In the PEKSrand-BG scheme, the protection is further enhanced in the sense that the adversary has to compromise both the gateway and at least

u proxies.

6.4.2 Privacy Analysis

As an extension of PEKS, both variants of the PEKSrand scheme inherit PEKS's capability of ensuring plaintext privacy. Hence, in this paper we limit privacy analysis to the protection of predicate privacy, more specifically, privacy protection against brute-force guessing attacks and statistical guessing attacks. In addition, since these two types of attacks require the knowledge of trapdoors, which is only held by the semi-trusted gateway and the receiver, in the following analysis we focus on privacy protection against the gateway.

- **Protection against Brute-force Guessing Attacks** The root cause of brute-force guessing attacks against the original PEKS scheme is that, a predicate represents a deterministic and direct mapping between the original keyword and a trapdoor. In both variants of the PEKSrand scheme, such a mapping is changed. More specifically, the mapping represented by a predicate is neither deterministic (i.e., the original keyword is randomized before the generation of the trapdoor), nor direct (i.e., the mapping between the original keyword and the trapdoor is indirect, although there exists a direct mapping between the randomized keyword and the trapdoor). As a result, they are robust against brute-force guessing attacks.

- **Protection against Statistical Guessing Attacks** Unlike brute-force guessing attacks, in statistical guessing attacks the adversary has extra knowledge of the statistical distribution of keywords. We observe that, in spite of the randomization of keywords before trapdoor generation, in PEKSrand-BG there exists an indirect mapping between the original keyword and the trapdoor that is generated from a randomized instance of the original keyword. And such a mapping can be revealed through first recording the history of trapdoor mapping during the keyword searching procedure and then comparing the frequency of a specific keyword, which is obtained from the extra knowledge of the statistical distribution of keywords, with the frequency that each trapdoor has been successfully matched. Hence, the PEKSrand-BG scheme is vulnerable to statistical guessing attacks.

Since it is not feasible to limit the keyword usage at the sender side, we consider to mitigate the observed matched frequency of trapdoors during keyword searching phase at the gateway side. In the PEKSrand-SG variant, each keyword is mapped to multiple trapdoors instead of one in PEKSrand-BG. Theoretically, the PEKSrand-SG scheme is also vulnerable to statistical guessing attacks, since the expansion from one-to-one mapping to one-to-many mapping is applied to all keywords. Therefore, for two keywords x_1 and x_2, if the frequency of x_1 is higher than that of x_2, in PEKSrand-SG the frequency of any trapdoor that is mapped to x_1 is still higher than the frequency of any trapdoor that is mapped to x_2. For example, suppose that statistically the frequencies of x_1 and x_2 are 20% and 10%, respectively. Assume that, a proxy farm consisting of five proxies is deployed and each ciphertext has been transformed two times before being forwarded to the gateway. Hence, each keyword is mapped to $C_5^2 = 10$ trapdoors, and statistically the frequency of any trapdoor derived from x_1 (i.e., 2%) is higher than that of any trapdoor derived from x_2 (i.e., 1%). However, intuitively. given the same number of total successful trapdoor matching[2], to distinguish two events with the statistical probability of 2% and 1%, respectively, is more difficult than to distinguish two events with the statistical probability of 20% and 10%, respectively. In the following, we seek the theoretical basis that supports such an intuition and estimate the effectiveness of this method.

We begin with the entropy analysis [34]. We first define the *mapping factor* F as

[2]In the theoretical analysis, we ignore unsuccessful trapdoor matching due to two reasons. The adversary's knowledge of the statistical distribution of keywords is defined in terms of all matched keywords. Moreover, we argue that taking unsuccessful trapdoor matching into consideration actually introduces noise to the statistical distribution and thus favor our goal of privacy protection.

the number of trapdoors that are mapped to a single keyword. Hence, F is equal to 1 and C_U^u in PEKSrand-BG and PEKSrand-SG, respectively, where U and u denote the number of proxies in the proxy farm and the number of times that a ciphertext is transformed with distinct inverses. Let $X = \{x_1, x_2, \ldots, x_k\}$ denote the set of all keywords that the receiver chooses. Let $T_{BG} = \{T_1, T_2, \ldots, T_k\}$ denote the set of trapdoors corresponding to X in the PEKSrand-BG scheme. Let p_i denote the probability that keyword x_i or the corresponding trapdoor T_i is used. Hence, the entropy of keywords X or trapdoors T_{BG} in the PEKSrand-BG scheme, denoted as E_{BG}, can be calculated according to Equation (2).

$$E_{BG} = -\sum_{i=1}^{k} p_i \log p_i \tag{2}$$

Let $T_{SG} = \{T_1^1, \ldots, T_1^F, T_2^1, \ldots, T_2^F, \ldots, T_k^1, \ldots, T_k^F\}$ denote the set of trapdoors corresponding to X in the PEKSrand-SG scheme. p_i^j denote the probability that trapdoor T_i^j is used. In the PEKSrand-SG scheme, each keyword is mapped to F trapdoors evenly. Thus, we have $p_i^j = \frac{p_i}{F}$. Consequently, the entropy of keywords X or trapdoors T_{SG} in the PEKSrand-SG scheme, denoted as E_{SG}, can be calculated as follows.

$$\begin{aligned} E_{SG} &= -\sum_{i=1}^{k}\sum_{j=1}^{F} p_i^j \log p_i^j = -\sum_{i=1}^{k}\sum_{j=1}^{F} \frac{p_i}{F} \log \frac{p_i}{F} \\ &= -\sum_{i=1}^{k} p_i \log \frac{p_i}{F} = -\sum_{i=1}^{k} p_i(\log p_i - \log F) \\ &= -\sum_{i=1}^{k} p_i \log p_i + \sum_{i=1}^{k} p_i \log F \\ &= E_{BG} + \sum_{i=1}^{k} p_i \log F = E_{BG} + \log F \end{aligned} \tag{3}$$

According to Equation (3), compared to PEKSrand-BG, the entropy of keywords X is improved by a value of $\log F$ in PEKSrand-SG. In addition, Equation (3) also shows that, by increasing the mapping factor F, we can achieve better privacy protection on keywords. In practice, compared to entropy, probability is a more intuitive representation of the privacy criteria. Hence, in the following we present the probability analysis so as to illustrate the trade-off between privacy and efficiency in a more clear and intuitive manner.

To perform further analysis, we introduce a new concept called *n-F undistinguishable*. Let n denote the total number of successful trapdoor matching. Given a pair of trapdoors T_A and T_B corresponding to keywords A and B, respectively, without loss of generality, we assume that the frequency of A is higher than that of B according to the statistical distribution of keywords. If the actual number of times that trapdoor T_B is matched is no less than that of trapdoor T_A, we say that "the trapdoor T_A is n-F undistinguishable from the trapdoor T_B". Hence, our design goal is to maximize the probability of n-F undistinguishable, denoted as p_{n-F}. Further, let p_{n-F}^{BG} and p_{n-F}^{SG} denote the probability of n-F undistinguishable in PEKSrand-BG and PEKSrand-SG, respectively.

If we view each trapdoor matching as an experiment with only two possible results (i.e., "T_A is matched" and "Otherwise"), the probability of trapdoor T_A is matched k_A times in a sequence of n independent matching experiments can be calculated using Equation (4) according to the binomial distribution.

$$p_{A-k} = f(n, k_A, p_A) = C_n^{k_A} p_A^{k_A} (1 - p_A)^{n-k_A} \tag{4}$$

where p_A is the statistical probability that trapdoor T_A is matched in the n independent experiments.

Similarly, if we view each trapdoor matching as an experiment with only two possible results (i.e., "T_B is matched" and "Otherwise"), among the remained $n - k_A$ independent matching experiments (namely, excluding k_A independent experiments matching T_A), the probability of trapdoor T_B is matched k_B times, given that trapdoor T_A is matched k_A times, can be calculated using Equation (5) according to the binomial distribution.

$$\begin{aligned} p_{B-k} &= f(n-k_A, k_B, \frac{n}{n-k_A} \cdot p_B) \\ &= C_{n-k_A}^{k_B} (\frac{n}{n-k_A} \cdot p_B)^{k_B} (1 - \frac{n}{n-k_A} \cdot p_B)^{n-k_A-k_B} \end{aligned} \tag{5}$$

where p_B is the statistical probability that trapdoor T_B is matched in terms of all n experiments including those experiments matching T_A. Hence, p_{n-F}^{BG} can be calculated as shown in Equation (6).

$$\begin{aligned} p_{n-F}^{BG} &= \sum_{k_A=0}^{n} [f(n, k_A, p_A) * \sum_{k_B=k_A}^{n-k_A} f(n-k_A, k_B, \frac{n}{n-k_A} \cdot p_B)] \\ &= \sum_{k_A=0}^{n} [C_n^{k_A} p_A^{k_A} (1-p_A)^{n-k_A} * \\ &\quad \sum_{k_B=k_A}^{n-k_A} C_{n-k_A}^{k_B} (\frac{n \cdot p_B}{n-k_A})^{k_B} (1 - \frac{n \cdot p_B}{n-k_A})^{n-k_A-k_B}] \end{aligned} \tag{6}$$

In the PEKSrand-SG, the one-to-one mapping between an original keyword and a corresponding trapdoor is transformed into a one-to-F mapping. Accordingly, the statistical probabilities of any trapdoor T'_A mapped to keyword A and any trapdoor T'_B mapped to keyword B are changed into $p'_A = \frac{p_A}{F}$ and $p'_B = \frac{p_B}{F}$, respectively. Hence, p_{n-F}^{SG} can be calculated according to Equation (7).

$$
\begin{aligned}
p^{SG}_{n-F} &= \sum_{k_A=0}^{n}[f(n,k_A,p'_A) * \sum_{k_B=k_A}^{n-k_A} f(n-k_A,k_B,\frac{n}{n-k_A}\cdot p'_B)] \\
&= \sum_{k_A=0}^{n}[C_n^{k_A}(\frac{p_A}{F})^{k_A}(1-\frac{p_A}{F})^{n-k_A} * \\
&\quad \sum_{k_B=k_A}^{n-k_A} C^{k_B}_{n-k_A}[\frac{np_B}{(n-k_A)F}]^{k_B}[1-\frac{np_B}{(n-k_A)F}]^{n-k_A-k_B}] \qquad (7)
\end{aligned}
$$

Let $p^{SG\prime}_{n-F}$ denote the probability of n-F undistinguishable in PEKSrand-SG, given any possible pair of trapdoors T_A and T_B corresponding to keywords A and B, respectively. According to Equation (8), we know that $p^{SG}_{n-F} = p^{SG\prime}_{n-F}$.

$$
p^{SG\prime}_{n-F} = \sum_{i=1}^{F}\sum_{j=1}^{F} p_{ij}\cdot p^{(i,j)}_{n-F} = \sum_{i=1}^{F}\sum_{j=1}^{F}\frac{1}{F^2}\cdot p^{SG}_{n-F} = p^{SG}_{n-F} \qquad (8)
$$

where p_{ij} denote that the probability of a specific pair that consists of the ith trapdoor corresponding to keyword A and the jth trapdoor corresponding to keyword B is chosen and $p^{(i,j)}_{n-F}$ denote the probability of n-F undistinguishable when such a pair is chosen.

In Figure 16, we compare the probabilities of n-F undistinguishable in PEKSrand-BG and PEKSrand-SG under four settings of p_A and p_B. As shown in Figure 16, p^{SG}_{n-F} is much higher than p^{BG}_{n-F} in all settings. Nonetheless, careful readers may notice that in Figure 16 the probability of n-F undistinguishable may drop to a low level if n is big enough, i.e., by observing a large number of trapdoor matching records, even in the PEKSrand-SG scheme.

To address this issue, we need to perform periodical secret refreshments, i.e., executing the setup phase in PEKSrand-SG after executing a predetermined number of

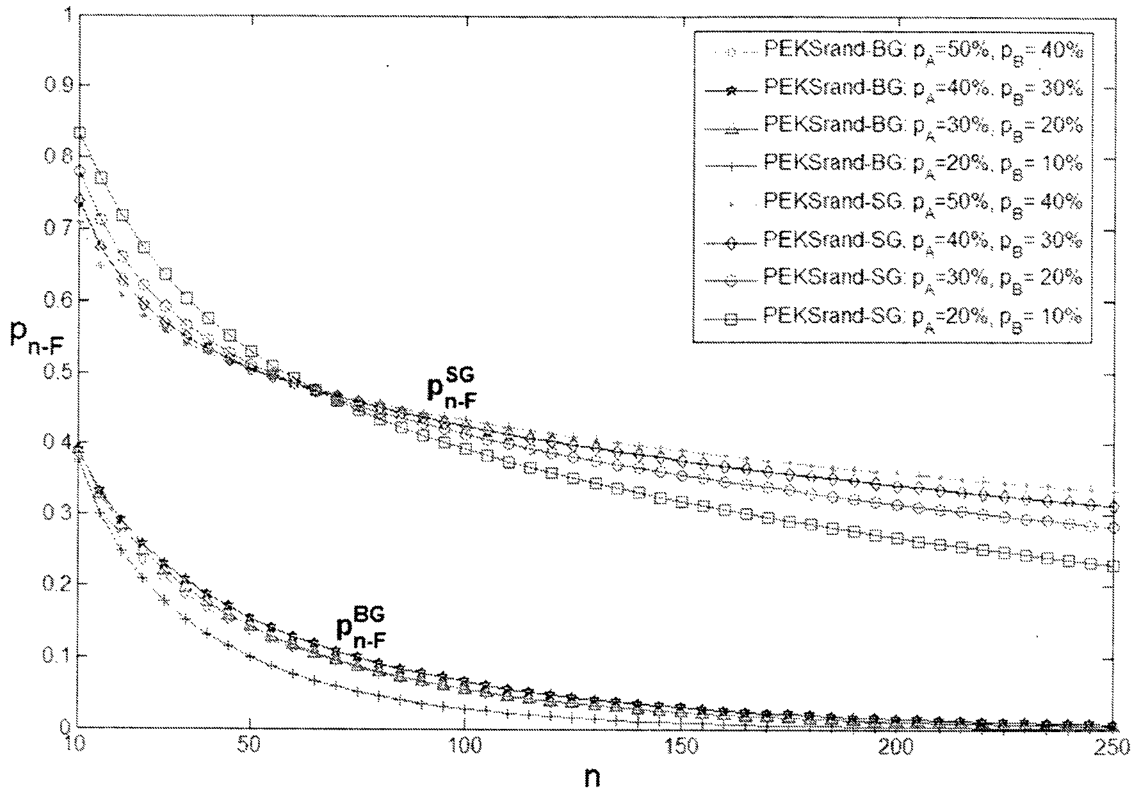


Figure 16: Probability Comparison of PEKSrand-BG and PEKSrand-SG (10–250)

successful trapdoor matchings, which we call the refreshing threshold. Apparently, there exists a trade-off between privacy and efficiency. More specifically, Figure 17 shows the trade-off between the probability of n-F undistinguishable and the refreshing threshold when $p_A = 30\%$ and $p_B = 20\%$. Note that, given that the number of proxies is fixed, to maximize the mapping factor F, we choose $u = \lfloor \frac{U}{2} \rfloor$. According to Figure 17, given a specific requirement on p^{SG}_{n-F}, by slightly increasing the number of proxies in the proxy farm, the refreshing threshold can be improved significantly. For example, assume that we set the privacy requirement as $p^{SG}_{n-F} \geq 0.5$, Table 7 shows the maximum refreshing threshold satisfying the privacy requirement under different settings of proxy farm and random walking. The mapping factor and the refreshing threshold is denoted as F and TD, respectively, in Table 7. When the setting of random walking is $(U,\ u) = (5,\ 2)$, the maximum refreshing threshold is

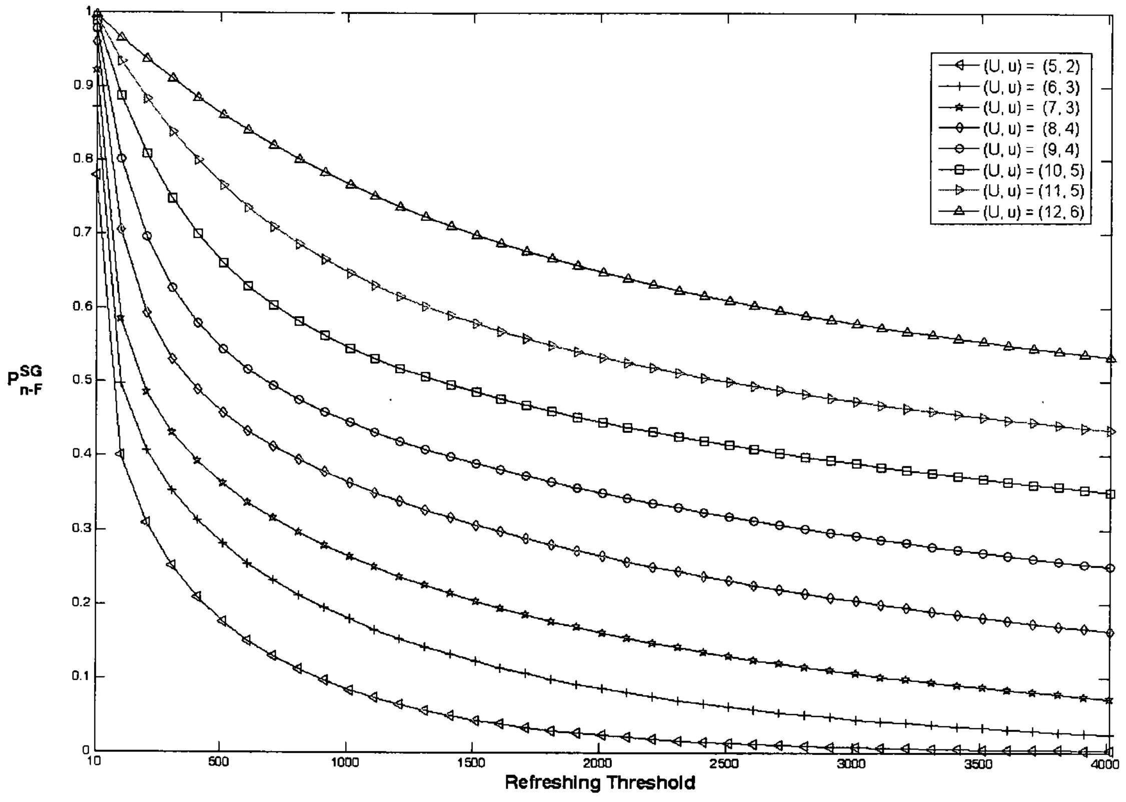


Figure 17: Tradeoff between Statistics Privacy and Storage Efficiency

only 53. By simply increasing the number of proxies to 10 and 11, the maximum refreshing threshold is increased to 1367 and 2507, respectively. We argue that, such refreshing thresholds are sufficient for many real applications, e.g., intelligent email routing. A more detailed analysis about the overhead of the proposed scheme is given in Section 6.5.

6.5 Efficiency Analysis and Empirical Results

Our PEKSrand implementation leverages the Identity Based Encryption [16] algorithms implemented in the MIRACL library [2]. We adopt the well-known Tate Pairing, which is the heart of the ciphertext generation, transformation, and testing processes. In our extension of PEKS, a 512-bit prime p is used for effective 1024-bit security, and G_1 and G_2

Table 7: Mapping Factors and Refreshing Thresholds When $p^{SG}_{n-F} \geq 0.5$

(U, u)	F	TD	TD/F
(5, 2)	10	53	5.30
(6, 3)	20	108	5.40
(7, 3)	35	189	5.40
(8, 4)	70	379	5.41
(9, 4)	126	683	5.42
(10, 5)	242	1367	5.65
(11, 5)	462	2507	5.43

are groups on the supersingular elliptic curve $y^2 = x^3 + x \bmod p$ with 160-bit group order $q = 2^{159} + 2^{17} + 1$, a prime which divides p.

We simulate the PEKSrand-BG and PEKSrand-SG schemes on a desktop with an Intel(R) Core (TM)2 2.13GHz CPU (64-bit processor) and 2GB RAM. The programs run on Windows XP Professional operation system with ADO database connection to a Microsoft SQL 2000 database server.

6.5.1 Computation Overhead

Compared to the original PEKS, in terms of computation, PEKSrand-BG introduces only one additional exponential calculation per ciphertext at the proxy. In PEKSrand-SG, similarly, each proxy involved in the random walking process performs only one additional exponential calculation per ciphertext.

Note that, the number of trapdoor matching that the gateway needs to perform in PEKSrand-SG is the same as that of the original PEKS. It is due to the fact that, the trapdoor matching process is limited to a subset that contains d elements, where d is the number of keywords that Alice chooses, in spite that the total number of trapdoors is increased by a factor of F. More specifically, the gateway needs to perform $\frac{d}{2}$ trapdoor matching operation on average. Hence, the only additional operation at the gateway is to identify the subset that should be subject to the following trapdoor matching operation.

On our testbed, an exponential multiplication and a trapdoor matching operation take 5 milliseconds and 13 milliseconds, respectively. And the operation of identifying a subset is very efficient. It takes only 15 milliseconds in the worst case (i.e., $U = 11$ and $F = 462$). Therefore, compared to the original PEKS scheme, the additional computation overhead introduced by the PEKSrand schemes is negligible. Moreover, given a reasonable size of keywords to be searched, the actual computation overhead is small in most real world applications.

Table 8 shows the simulation results about the performance of the original PEKS and two PEKSrand schemes. The sample set we used are 256 keywords extracted from the Enron Email Dataset [43]. For each round, we randomly chose 50 out of 256 keywords and encrypt them, and then record both the number of trapdoor matching operations performed, as well as the exact time used at the gateway. Let $\#_{Test}$ and T_{Time} denote the number of trapdoor matching operations performed to identify all 50 keywords chosen and the time used that complete the keyword matching process. The results shown in the Table 8 are the averages of 50 rounds.

Table 8: Simulation Results

Scheme	$\#_{Test}$	T_{Time} (Second)
PEKS	6510	88.61
PEKSrand-BG	6510	88.61
PEKSrand-SG	6510	88.63

6.5.2 Storage Overhead

PEKSrand-BG has the same storage overhead at the gateway, since the number of trapdoors assigned to the gateway in PEKSrand-BG is the same as that of PEKS. In contrast, in PEKSrand-SG, the storage overhead is increased by a factor of F, while the original one-to-one mapping in PEKS is converted into a one-to-F mapping.

In our implementation, to achieve effective 1024-bit security, the size of a trapdoor is 128 bytes. Thus, given that there are 256 keywords in our simulation, the total storage overheads of PEKSrand-BG and PEKSrand-SG at the gateway's side are 32768 bytes and 32768*F bytes, respectively. We argue that, the storage overhead of PEKSrand-SG is still acceptable in many real world applications, considering that nowadays it is common that the hard drives of a server have the capacity of 1TB or more [1]. For example, in the intelligent email routing application, gateway with 1TB storage can support more than 2^{13} users, given that $F = 462$ and the number of keywords that each user specifies is 2^8 on average.

6.5.3 Communication Overhead

If we view the proxy in PEKSrand-BG or the proxy farm in PEKSrand-SG as a transparent component between the sender and the gateway, there is actually no additional traffic generated in the PEKSrand schemes, since the same number and size of ciphertexts are transmitted, although the content of packets are changed. However, the PEKSrand schemes indeed introduce some delay due to the ciphertext transformation and random walking within the proxy farm. Fortunately, as shown in Section 6.5.1, the ciphertext transformation operation is lightweight. In addition, analysis in Section 6.4.2 (in particular Table 7) shows that only a small u, e.g., 4 or 5, is sufficient to satisfy the privacy requirement of most real world applications. Consequently, compared to the original PEKS scheme, the additional communication overhead introduced by PEKSrand-BG and PEKSrand-SG is negligible.

Chapter 7

Conclusion and Future Work

In this chapter, we finalize this thesis by drawing conclusions and proposing future works.

7.1 Conclusion

Nowadays, intrusion detection systems (IDSs) are widely deployed into hosts and networks as a second line of defense to defend against increasing number of various attacks. IDSs generate intrusion detection alerts when suspicious activities are observed. Although numerous new techniques have been proposed and adopted to build better intrusion detection systems, current IDSs still have three well-known problems: (1) generated IDS alerts usually contain a lot of false positive or false negative, (2) IDSs usually produce huge number of alerts which overwhelm network administrators, (3) a single IDS is not capable to detect large-scale or cross-domain multi-hop attacks.

IDS alert correlation techniques are thus proposed to mitigate these IDS troubles. These techniques have been proven efficiently as well as effectively and able to help analysts better understand security threats further to take appropriate responses. However, privacy concerns, such as sensitive information contained in the shared security alerts may be disclosed and misused, stopped IDS alert correlation systems especially large-scale cross-domain

IDS alert correlation systems deployed in the reality.

This thesis focuses on cross-domain intrusion detection alert correlation. In particular, we proposed solutions for the following three problems.

- **Privacy-Preserving Alert Correlation** To address privacy concerns, it is always highly desirable and usually mandatory to anonymize sensitive attributes in IDS alerts before sharing. Previous proposed sanitization techniques may work well on some particular circumstances by seeking a tradeoff between privacy and usabiltiy. However, these most commonly used anonymization techniques have huge negative effects on delicate analysis of anonymized data (e.g., intrusion detection alert correlation).

  In this thesis, we proposed the first work that used Public-key Encryption with Keyword Search (PEKS) techniques to preserve data confidentiality without loose of data usability. We also gave a concrete system example, which operates according to the defined TEIRESIAS protocol, based on Ning et.al' alert correlation algorithm [51] to instantiate our idea. From our experiment results, we achieved ideal security and privacy outlooks.

- **Privacy-Preserving Report Retrieval** Another issue largely ignored in previous works in privacy-preserving alert sharing is fairness. Many previous solutions work well under an implicit assumption that, participants are motivated to share their data in exchange of more accurate analysis results and thus more effective defense. However, selfish users who also exist in reality may reluctant to provide any useful information but to share the final analysis results. Thus, fairness issue should not be ignored when designs a realistic system.

  In this thesis, we proposed a report retrieval mechanism, which is based on Symmetric-key Encryption with Keyword Search (SEKS), to address the fairness issue. The fairness is achieved through allowing contributors to request correlated results back if attack happens while selfish users are not able to get any information. Once again, the retrieval mechanism is privacy-preserving by guarding that any private information will not be disclosed during the whole retrieval procedure.

- **Predicate Privacy and Statistics Privacy in PEKS** The development of PEKS boosts many useful applications such as secure searchable automated Remote email storage [7] and privacy-preserving alert correlation presented in this thesis. However, recent research on attacks against PEKS [3, 8, 17, 18, 60] may discourage the usages of PEKS in real world applications.

  In this thesis, we identified a new type of attacks against the original PEKS scheme (i.e., statistical guessing attacks) and proposed the PEKSrand scheme that aims at protecting predicate privacy and statistics privacy, which is a new concept introduced by us. Both variants of the PEKSrand scheme can prevent brute-force guessing attacks. However, only the PEKSrand-SG scheme can be used to mitigate statistical guessing attacks at the cost of a higher storage overhead at the gateway or delegate. According to our analysis and experimental results, both schemes introduce negligible additional communication and computation overheads and can be smoothly deployed in existing systems. The PEKSrand scheme can either be used independently or be combined with TEIRESIAS to further improve its privacy protection at the cost of the storage overhead at the correlator.

7.2 Future Work

Though in this thesis we have addressed a few problems in privacy-preserving intrusion detection alert correlation and implemented a workable system, there are still many improvements that can be developed based on the current results. In the following, we list two problems deserving further studies.

- **Support Online Analysis** Our current implementation is an offline alert correlation analysis system, which operates on anonymized alerts stored in the database. To support online analysis, a correlator should have real-time abilities to efficiently analyze all incoming alerts.

  Since the most expensive time consumption in TEIRESIAS is the calculation of bilinear pairings, and our current implementation is based on the software computation of bilinear pairings. So, one possible solution to enhance the efficiency of our system is by using hardware implementations of bilinear pairings. Moreover, all the calculation of time-consuming cryptographic primitives can also be replaced by hardware implementations.

  Current implementation of alert correlator is a DBMS-based application, which provides great convenience and reliable supports to manage IDS alerts. However, interaction with the DBMS is required to process each single alert, which introduces substantial performance overhead for a real time system. So, another possible approach to enhance the efficiency of our system is to draw support from memory management or query optimization techniques.

- **New Privacy-Preserving while Usable Techniques** An ideal privacy-preserving while usable technique should be able to provide strong privacy protection to all information as well as maintain support for various analysis based on anonymized data. In this thesis, the techniques we adopted are able to provide exact-matching test ability on the anonymized alert attributes. However, rich operations such as *Conjunctive, Subset, and Range* test ability are also highly desired in the cross-domain alert correlation circumstance in order to support both delicate analysis and roughly statistical analysis.

Bibliography

[1] Ibm system storage product guide. ftp://ftp.software.ibm.com/common/ssi/pm/rg/n/tso00364usen/TSO00364USEN.PDF.

[2] Multiprecision integer and rational arithmetic C/C++ library (MIRACL). http://www.shamus.ie/.

[3] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 21(3):350–391, July 2008.

[4] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5(10), September 2000. http://www.firstmonday.dk/issues/issue5_10/adar/.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 439 – 450, 2000.

[6] Mark Allman, Ethan Blanton, Vern Paxson, and Scott Shenker. Fighting coordinated attackers with cross-organizational information sharing. In *Proceedings of the Fifth Workshop on Hot Topics in Networks (HotNets-V)*, pages 121–126, 2006.

[7] Adam J. Aviv, Michael E. Locasto, Shaya Potter, and Angelos D. Keromytis. SSARES: Secure searchable automated remote email storage. In *ACSAC*, pages 129–139. IEEE Computer Society, 2007.

[8] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. Technical Report 2005/191, Cryptology ePrint Archive, 2005.

[9] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Proceedings of Advances in Cryptology – CRYPTO '96, LNCS 1109*, pages 1–15, 1996.

[10] John Bethencourt, Jason Franklin, and Mary Vernon. Mapping internet sensors with probe response attacks. In *Proceedings of the 14th USENIX Security Symposium*, pages 193–208, 2005.

[11] Joachim Biskup and Ulrich Flegel. On pseudonymization of audit data for intrusion detection. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009*, pages 161–180, 2000.

[12] Joachim Biskup and Ulrich Flegel. Transaction-based pseudonyms in audit data for privacy respecting intrusion detection. In *RAID 2000, LNCS 1907*, pages 28–48, 2000.

[13] Burton H. Bloom. Space/Time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422 – 426, July 1970.

[14] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of Advances in Cryptology (CRYPTO 2001), Lecture Notes in Computer Science 2139*, pages 213–229, 2001.

[15] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Proceedings of Eurocrypt 2004, LNCS 3027*, pages 506–522, 2004.

[16] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, June 2003.

[17] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.

[18] Jin Wook Byun, Hyun Suk Rhee, Hyun-A Park, and Dong Hoon Lee. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Secure Data Management, Third VLDB Workshop, SDM 2006, Seoul, Korea, September 10-11, 2006, Proceedings*, volume 4165 of *Lecture Notes in Computer Science*, pages 75–83. Springer, 2006.

[19] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam Smith, and Hoeteck Wee. Toward privacy in public databases. In *Proceedings of Theory of Cryptography (TCC'05), LNCS 3378*, pages 363–385, 2005.

[20] F. Cuppens. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17th Annual Computer Security Applications Conference*, 2001.

[21] F. Cuppens and R. Ortalo. Lambda: A language to model a database for detection of attacks. In *Proceedings of Recent Advances in Intrusion Detection (RAID 2000)*, 2000.

[22] O. Dain and R.K. Cunningham. Fusing a heterogeneous alert stream into scenarios.

[23] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[24] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *Proceedings of the 2001 Workshop on New Security Paradigms Workshop*, pages 13 – 22, 2001.

[25] S.T. Eckmann, G. Vigna, and R.A. Kemmerer. Statl: An attack language for state-based intrusion detection. *Journal of Computer Security*, 10, 2002.

[26] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '03)*, pages 211 – 222, 2003.

[27] Jinliang Fan, Jun Xu, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks*, 46:253–272, 2004.

[28] Ulrich Flegel. Pseudonymizing unix log files. In *Proceedings of International Conference on Infrastructure Security (InfraSec 2002), LNCS 2437*, pages 162–179, 2002.

[29] S. Frankel and H. Herbert. The AES-XCBC-MAC-96 algorithm and its use with IPsec. http://www.ietf.org/rfc/rfc3566.txt, September 2003. RFC 3566.

[30] Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, pages 193–206, 2002.

[31] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL protocol version 3.0. http://wp.netscape.com/eng/ssl3/draft302.txt, November 1996.

[32] Eu-Jin Goh. Secure indexes. http://eprint.iacr.org/, 2004. Cryptology ePrint Archive, Report 2003/216.

[33] Oded Goldreich. *Secure Multi-Party Computation*, October 2002. Working draft, Version 1.4.

[34] R. M. Gray. *Entropy and information theory.* New York, Springer Verlag, 1990.

[35] Philip Gross, Janak Parekh, and Gail Kaiser. Secure "selecticast" for collaborative intrusion detection systems. In *Proceedings of International Workshop on Distributed Event-Based Systems (DEBS '04)*, 2004.

[36] Guofei Gu, Alvaro A. Cardenas, and Wenke Lee. Principled reasoning and practical applications of alert fusion in intrusion detection systems. In *Proceedings of the 2006 ACM Symposium on Information, Computer, and Communication Security (ASIACCS'08)*, March 2008.

[37] L. T. Heberlein and S. Staniford-Chen. Holding intruders accountable on the internet. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 39–49, Oakland, CA, 1995.

[38] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

[39] IBM. IBM RealSecure server sensor. http://www-935.ibm.com/services/us/iss/pdf/realsecure-server-sensor-ss.pdf.

[40] Junjie Jiang. Bilinear pairing (Eta_T pairing) IP core. http://www.cs.cityu.edu.hk/ẽcc/doc/etat_datasheet_v2.pdf, May 2007. Data Sheet (Version 2.0).

[41] K. Julisch. Mining alarm clusters to improve alarm handling efficiency. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, pages 12–21, 2001.

[42] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *The 8th ACM International Conference on Knowledge Discovery and Data Mining*, 2002.

[43] Bryan Klimt and Yiming Yang. The Enron corpus: A new dataset for email classification research. In *15th European Conf. Machine Learning*, pages 217–226. Springer-Verlag, 2004. Lect. Notes Comput. Sci. vol. 3201.

[44] Tadayoshi Kohno, Andre Broido, and K.C. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[45] Adam J. Lee, Parisa Tabriz, and Nikita Borisov. A privacy-preserving interdomain audit framework. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society (WPES '06)*, pages 99 – 108, 2006.

[46] Patrick Lincoln, Phillip Porras, and Vitaly Shmatikov. Privacy-preserving sharing and correlation of security alerts. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[47] Michael E. Locasto, Janak J. Parekh, Angelos D. Keromytis, and Salvatore J. Stolfo. Towards collaborative security and P2P intrusion detection. In *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, pages 30–36, 2005.

[48] Emilie Lundin and Erland Jonsson. Privacy vs intrusion detection analysis. In *Recent Advances in Intrusion Detection (RAID)*, 1999.

[49] Emilie Lundin and Erland Jonsson. Anomaly-based intrusion detection: Privacy concerns and other problems. *Computer Networks*, 34:623–640, 2000.

[50] B.C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, 1994.

[51] Peng Ning, Yun Cui, and Douglas S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, pages 245–254, 2002.

[52] Office of Inspector General. Homeland security information network could support information sharing more effectively. Technical Report OIG-06-38, Department of Homeland Security, June 2006.

[53] Ruoming Pang and Vern Paxson. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03)*, pages 339 – 351, 2003.

[54] Ruoming Pangy, Mark Allmanz, Vern Paxsonz, and Jason Lee. The devil and packet trace anonymization. *ACM SIGCOMM Computer Communication Review*, 36(1):29 – 38, January 2006.

[55] Janak J. Parekh, Ke Wang, and Salvatore J. Stolfo. Privacy-preserving payload-based correlation for accurate malicious traffic detection. In *Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense (LSAD '06)*, pages 99 – 106, 2006.

[56] J.-F. Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *DIAUqŕ00, Lecture Notes in Computer Science 2009*, pages 10–29, 2000.

[57] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: K-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98-04, SRI Computer Science Laboratory, 1998.

[58] Michael Scott. Computing the tate pairing. In *Proceedings of Topics in Cryptology – CT-RSA 2005, LNCS 3376*, pages 293–304, 2005.

[59] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[60] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2009.

[61] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In *ACM Conference on Computer and Communications Security (CCS 2000)*, pages 33–42, 2000.

[62] Yoichi Shinoda, Ko Ikai, and Motomu Itoh. Vulnerabilities of passive internet threat monitors. In *Proceedings of the 14th USENIX Security Symposium*, pages 209–224, 2005.

[63] Vitaly Shmatikov and Ming-Hsiu Wang. Security against probe-response attacks in collaborative intrusion detection. In *Proceedings of the 2007 Workshop on Large Scale Attack Defense (LSAD'07)*, pages 129–136, 2007.

[64] Adam Slagell, Kiran Lakkaraju, and Katherine Luo. FLAIM: A multi-level anonymization framework for computer and network logs. In *Proceedings of the 20th USENIX Large Installation System Administration Conference (LISA '06)*, 2006.

[65] Adam Slagell, Jun Wang, and William Yurcik. Network log anonymization: Application of crypto-PAn to cisco netflows. In *NSF/AFRL Workshop on Secure Knowledge Management (SKM qŕ04)*, 2004.

[66] Adam J Slagell, Yifan Li, and Katherine Luo. Sharing network logs for computer forensics: A new tool for the anonymization of NetFlow records. In *Computer Network Forensics Research (CNFR) Workshop*, 2005.

[67] Michael Sobirey, Simone Fischer-Hübner, and Kai Rannenberg. Pseudonymous audit for privacy enhanced intrusion detection. In *Proceedings of the IFIP TC11 13th International Conference on Information Security (SECqŕ97)*, pages 151–163, 1997.

[68] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of IEEE Symposium on Security and Privacy (S&P 2000)*, pages 44–55, 2000.

[69] Symantec. Symantec internet security threat report: Trends for january–june 07, September 2007.

[70] S. Templeton and K. Levitt. A requires/provides model for computer attacks.

[71] E. Totel, B. Vivinis, and L. Me. A language driven ids for event and alert correlation.

[72] A. Valdes and K. Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, pages 54–68, 2001.

[73] F. Valeur, G. Vigna, C.Kruegel, and R. Kemmerer. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3), 2004.

[74] Ke Wang, Gabriela Cretu, and Salvatore J. Stolfo. Anomalous payload-based worm detection and signature generation. In *Proceedings of Symposium on Recent Advances in Intrusion Detection (RAID), LNCS 3858*, pages 227–246, 2005.

[75] Ke Wang, Janak J. Parekh, and Salvatore J. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Proceedings of Symposium on Recent Advances in Intrusion Detection (RAID), LNCS 4219*, pages 226–248, 2006.

[76] Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. In *Proceedings of Symposium on Recent Advances in Intrusion Detection (RAID), LNCS 3224*, pages 203–222, 2005.

[77] Xinyuan Wang and Douglas S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS-03)*, pages 20–29.

[78] Dingbang Xu and Peng Ning. Privacy-preserving alert correlation: A concept hierarchy based approach. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005)*, 2005.

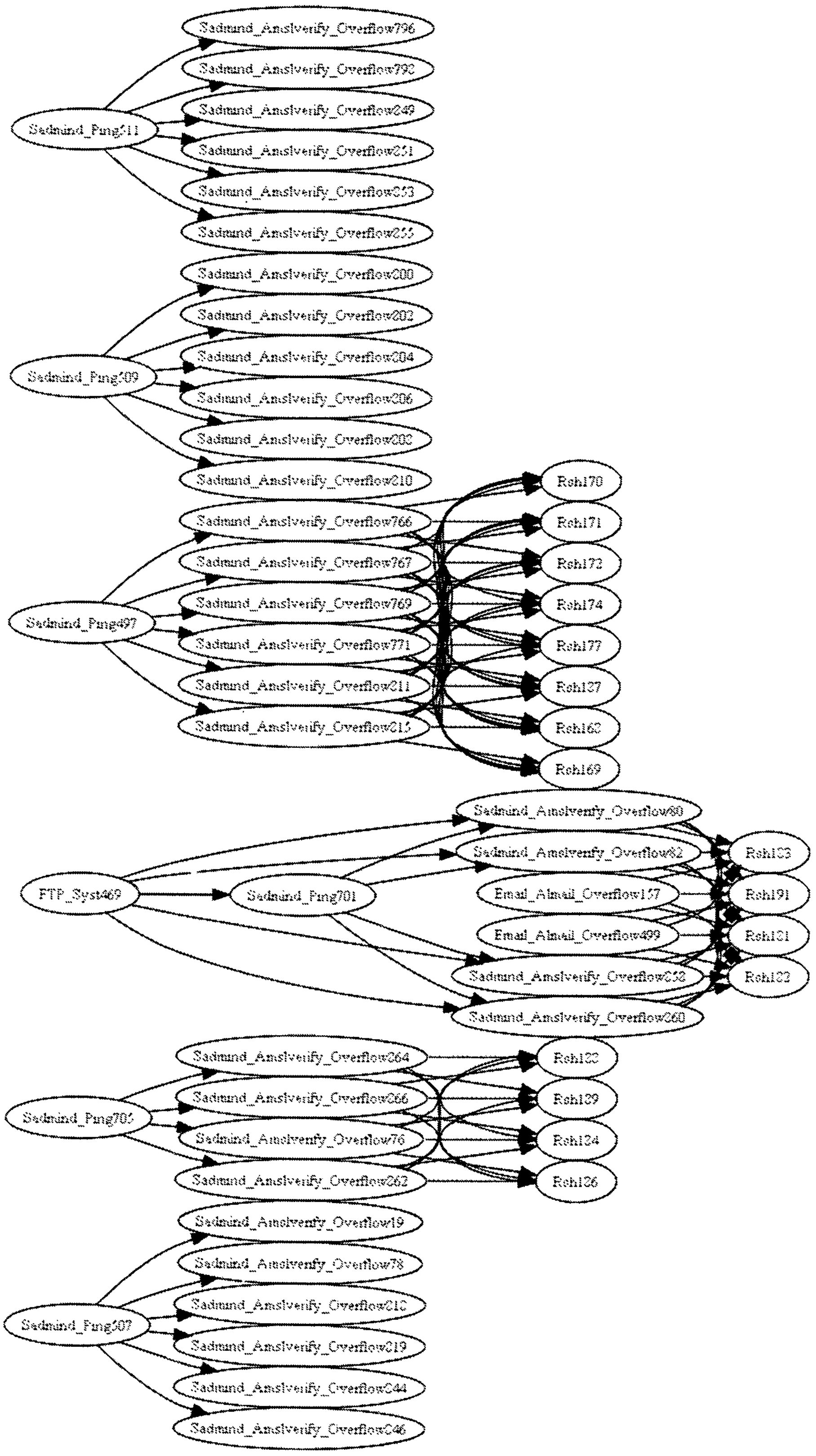
[79] Dingbang Xu and Peng Ning. A flexible approach to intrusion alert anonymization and correlation. In *Proceedings of 2nd IEEE Communications Society/CreateNet International Conference on Security and Privacy in Communication Networks (SecureComm 2006)*, 2006.

[80] Jun Xu, Jinliang Fan, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *Proceedings of the 10 Th IEEE International Conference on Network Protocols (ICNPąŕ02)*, pages 280 – 289, 2002.

[81] Yin Zhang and Vern Paxson. Detecting stepping stones. In *Proceedings of the 9th USENIX Security Symposium*, 2000.

Appendix A

LLDOS 1.0 DMZ Dataset Correlation Result



Sadmind_Ping511
Sadmund_Amslverify_Overflow796
Sadmund_Amslverify_Overflow792
Sadmund_Amslverify_Overflow249
Sadmund_Amslverify_Overflow251
Sadmund_Amslverify_Overflow253
Sadmund_Amslverify_Overflow255
Sadmind_Ping509
Sadmund_Amslverify_Overflow200
Sadmund_Amslverify_Overflow202
Sadmund_Amslverify_Overflow204
Sadmund_Amslverify_Overflow206
Sadmund_Amslverify_Overflow208
Sadmund_Amslverify_Overflow210
Sadmind_Ping497
Sadmund_Amslverify_Overflow766
Sadmund_Amslverify_Overflow767
Sadmund_Amslverify_Overflow769
Sadmund_Amslverify_Overflow771
Sadmund_Amslverify_Overflow211
Sadmund_Amslverify_Overflow215
Rsh170
Rsh171
Rsh172
Rsh174
Rsh177
Rsh127
Rsh162
Rsh169
FTP_Syst469
Sadmind_Ping701
Sadmind_Amslverify_Overflow80
Sadmind_Amslverify_Overflow82
Email_Almail_Overflow157
Email_Almail_Overflow499
Sadmund_Amslverify_Overflow258
Sadmund_Amslverify_Overflow260
Rsh103
Rsh191
Rsh131
Rsh122
Sadmind_Ping705
Sadmund_Amslverify_Overflow264
Sadmund_Amslverify_Overflow266
Sadmind_Amslverify_Overflow76
Sadmund_Amslverify_Overflow262
Rsh122
Rsh139
Rsh124
Rsh126
Sadmind_Ping507
Sadmind_Amslverify_Overflow19
Sadmind_Amslverify_Overflow78
Sadmund_Amslverify_Overflow210
Sadmund_Amslverify_Overflow219
Sadmund_Amslverify_Overflow244
Sadmund_Amslverify_Overflow246

Figure 18: LLDOS 1.0 DMZ Dataset Correlation Result

Appendix B

LLDOS 1.0 Inside Dataset Correlation Result

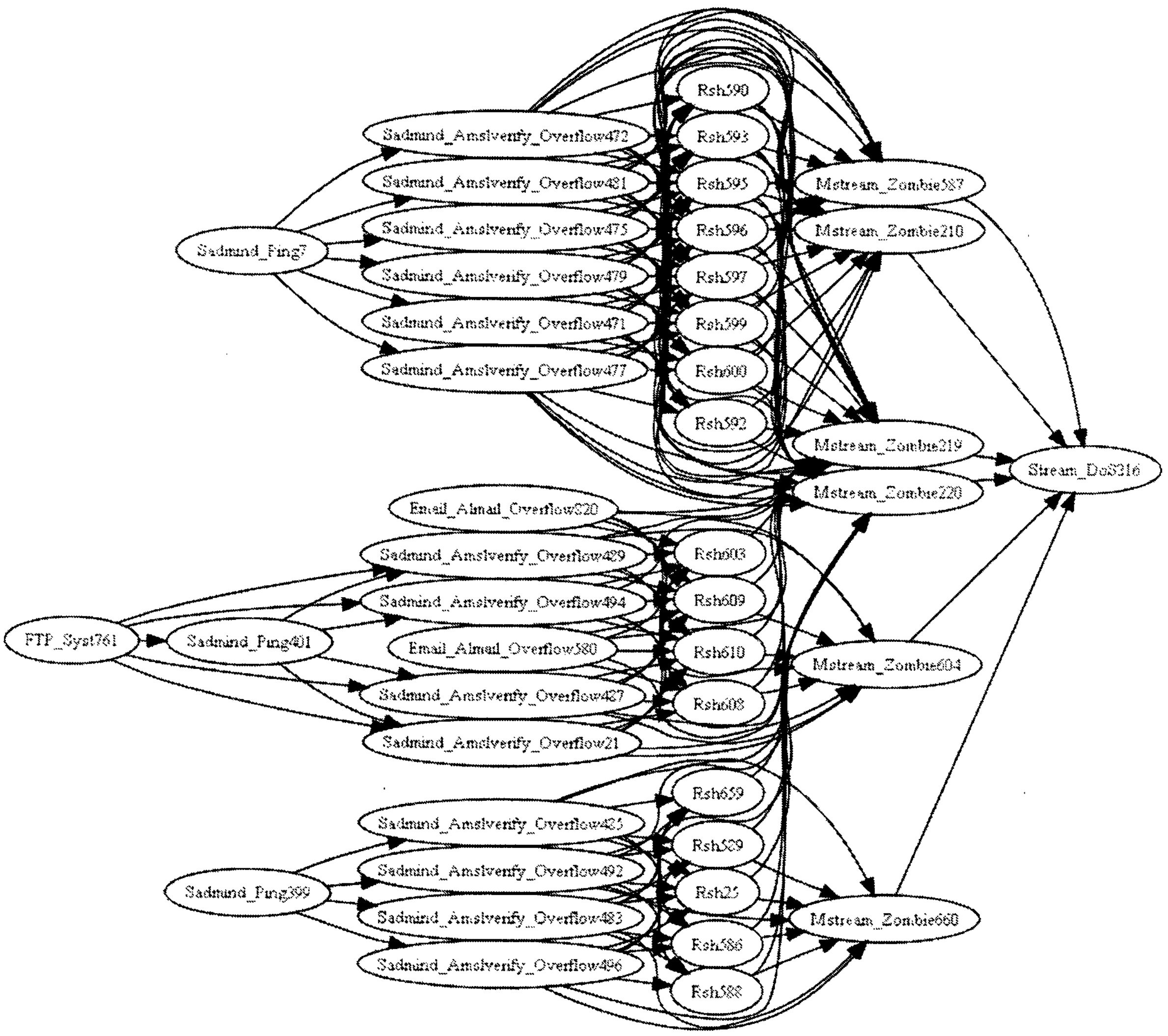

Sadmind_Ping7
Sadmind_Amslverify_Overflow472
Sadmind_Amslverify_Overflow481
Sadmind_Amslverify_Overflow475
Sadmind_Amslverify_Overflow479
Sadmind_Amslverify_Overflow471
Sadmind_Amslverify_Overflow477
Rsh590
Rsh593
Rsh595
Rsh596
Rsh597
Rsh599
Rsh600
Rsh592
Mstream_Zombie587
Mstream_Zombie210
Mstream_Zombie219
Mstream_Zombie220
Stream_DoS316
Email_Almail_Overflow820
Sadmind_Amslverify_Overflow429
Sadmind_Amslverify_Overflow494
FTP_Syst761
Sadmind_Ping401
Email_Almail_Overflow580
Sadmind_Amslverify_Overflow427
Sadmind_Amslverify_Overflow21
Rsh603
Rsh609
Rsh610
Rsh608
Mstream_Zombie604
Rsh659
Rsh529
Rsh25
Rsh586
Rsh588
Sadmind_Ping399
Sadmind_Amslverify_Overflow485
Sadmind_Amslverify_Overflow492
Sadmind_Amslverify_Overflow483
Sadmind_Amslverify_Overflow496
Mstream_Zombie660

Figure 19: LLDOS 1.0 Inside Dataset Correlation Result

Appendix C

LLDOS 1.0 Cross-domain Correlation Result

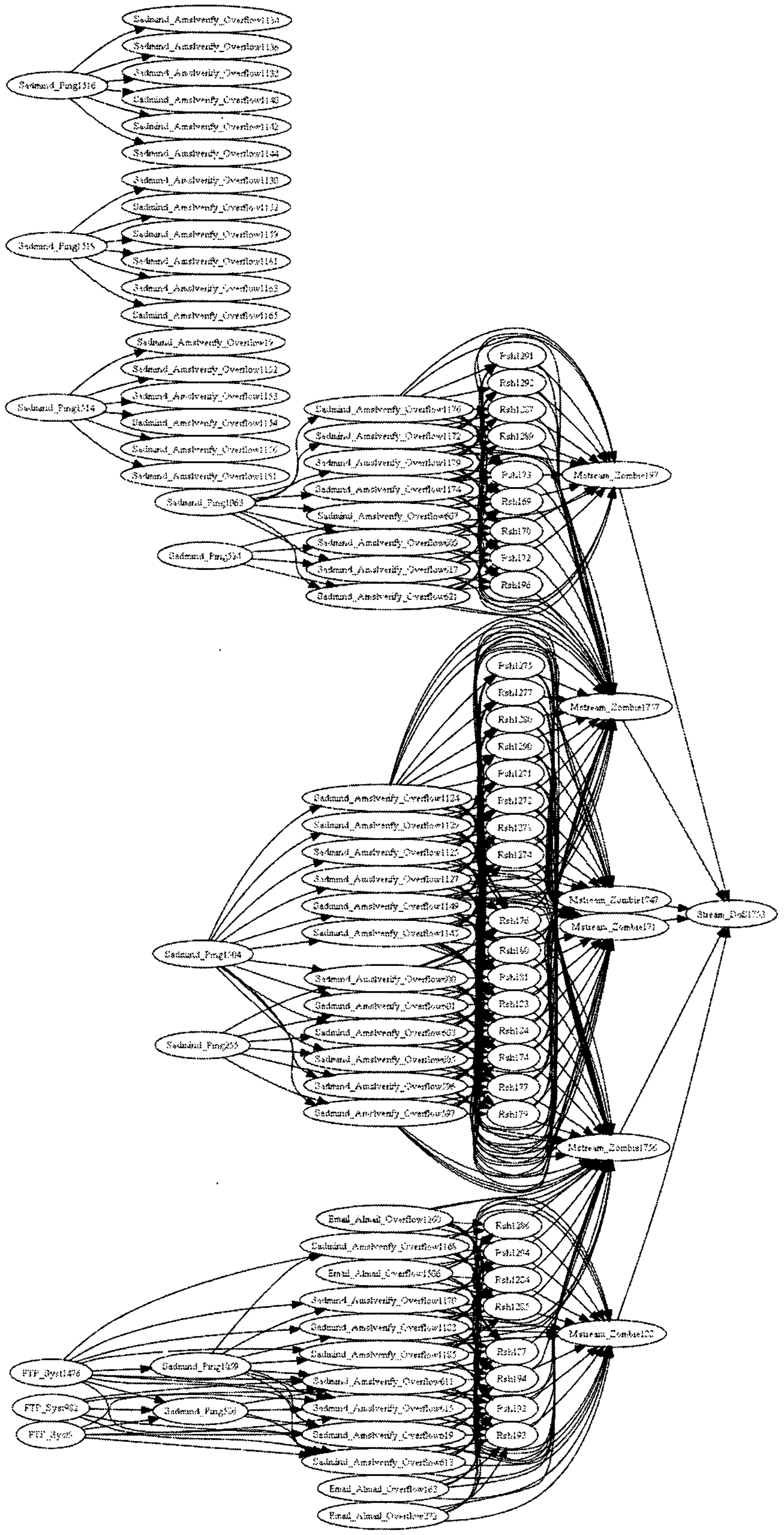

Sadmind_Ping1516
Sadmind_Amslverify_Overflow1134
Sadmind_Amslverify_Overflow1136
Sadmind_Amslverify_Overflow1132
Sadmind_Amslverify_Overflow1140
Sadmind_Amslverify_Overflow1142
Sadmind_Amslverify_Overflow1144
Sadmind_Ping1515
Sadmind_Amslverify_Overflow1130
Sadmind_Amslverify_Overflow1133
Sadmind_Amslverify_Overflow1159
Sadmind_Amslverify_Overflow1161
Sadmind_Amslverify_Overflow1163
Sadmind_Amslverify_Overflow1165
Sadmind_Ping1514
Sadmind_Amslverify_Overflow19
Sadmind_Amslverify_Overflow1152
Sadmind_Amslverify_Overflow1153
Sadmind_Amslverify_Overflow1154
Sadmind_Amslverify_Overflow1176
Sadmind_Amslverify_Overflow1151
Sadmind_Ping1063
Sadmind_Ping534
Sadmind_Amslverify_Overflow1176
Sadmind_Amslverify_Overflow1172
Sadmind_Amslverify_Overflow1179
Sadmind_Amslverify_Overflow1174
Sadmind_Amslverify_Overflow607
Sadmind_Amslverify_Overflow609
Sadmind_Amslverify_Overflow617
Sadmind_Amslverify_Overflow621
Rsh1291
Rsh1292
Rsh1287
Rsh1289
Rsh173
Rsh169
Rsh170
Rsh172
Rsh196
Mstream_Zombie197
Rsh1275
Rsh1277
Rsh1286
Rsh1290
Rsh1271
Rsh1272
Rsh1273
Rsh1274
Mstream_Zombie1737
Sadmind_Amslverify_Overflow1124
Sadmind_Amslverify_Overflow1126
Sadmind_Amslverify_Overflow1125
Sadmind_Amslverify_Overflow1127
Sadmind_Amslverify_Overflow1149
Sadmind_Amslverify_Overflow1147
Sadmind_Ping1504
Sadmind_Ping255
Sadmind_Amslverify_Overflow590
Sadmind_Amslverify_Overflow601
Sadmind_Amslverify_Overflow603
Sadmind_Amslverify_Overflow595
Sadmind_Amslverify_Overflow596
Sadmind_Amslverify_Overflow597
Rsh176
Rsh160
Rsh161
Rsh163
Rsh164
Rsh174
Rsh177
Rsh179
Mstream_Zombie1747
Mstream_Zombie171
Stream_DoS1753
Mstream_Zombie1756
Email_Almail_Overflow1269
Sadmind_Amslverify_Overflow1168
Email_Almail_Overflow1506
Sadmind_Amslverify_Overflow1170
Sadmind_Amslverify_Overflow1102
Sadmind_Amslverify_Overflow1185
Sadmind_Amslverify_Overflow611
Sadmind_Amslverify_Overflow615
Sadmind_Amslverify_Overflow619
Sadmind_Amslverify_Overflow613
Email_Almail_Overflow163
Email_Almail_Overflow272
Rsh1286
Rsh1294
Rsh1204
Rsh1285
Rsh177
Rsh194
Rsh192
Rsh193
Mstream_Zombie100
FTP_Syst1476
FTP_Syst982
FTP_Syst6
Sadmind_Ping1509
Sadmind_Ping526

Figure 20: LLDOS 1.0 Cross-domain Correlation Result