TOWARDS SENDER ACCOUNTABILITY ON EMAIL INFRASTRUCTURE USING SENDER IDENTITY AND REPUTATION MANAGEMENT

by

Gautam Singaraju

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Information Technology

Charlotte

2009

Approved by:

____
Dr. Brent Kang

____
Dr. Anita Raja

____
Dr. Xintao Wu

____
Dr. Jack Cathey

ABSTRACT

GAUTAM SINGARAJU. Towards sender accountability on email infrastructure using sender identity and reputation management. (Under the direction of DR. BRENT BYUNGHOON KANG)

Email Infrastructure has grown exponentially, since the early days of ARPANET, to support millions of users. However, the extensive adoption of the original open design has led to security implications. As claimed in recent statistics, about 95% of the emails are unsolicited and place phishing losses at $500 million.

Even though, current email-filtering technologies weed out most of the incoming spam, there is a need to hold senders accountable for their email behavior. Without sender accountability, there is no way to hold senders responsible for their online email behavior. Holding senders accountable helps identify senders who propagate spam, and possibly reduce the spam transmitted.

Holding a sender accountable for the sender's online activity requires: first, the sender's identification; and second, maintenance of its historical email activity. Today, widely deployed sender identity techniques counteract email spoofing by authenticating the sender's email server to the receiver organizations. Unfortunately, these techniques are not as effective as originally intended as: a) the senders create their own identity; b) spam-propagating senders have adopted these technologies.

Knowledge of the sender's identity alone does not guarantee its adherence to email best practices. Towards establishing sender accountability, this dissertation proposes RepuScore, a collaborative reputation framework that allows participating receiver organizations to share sender's behavioral patterns. In addition, this dissertation also explores Privilege Messaging (P-Messaging) framework, a fine-granular sender-authorization framework where each sender holds a set of credentials (privileges) to send an email; the receivers verify the attached credentials before accepting the emails. P-Messaging attempts to maintain trust among organizations with the help of a central authority, which periodically verifies the participating organization's adherence to good email practices.

To create a long-standing history, participating organizations locally collect information about the senders - from users or existing spam classification mechanisms that are submitted to a central RepuScore authority - to compute a global reputation summary. This dissertation discusses the distributed architecture and the algorithms designed to compute reputation based on the sender's a) spam rate (RepuScore) or b) spam rate and email volume (Volume-Enhanced RepuScore).

Additionally, the dissertation shares findings from experiments based on a RepuScore prototype using a) simulation logs; and b) deployed SpamAssassin plug-in since 10/9/2007 at three organizations. Based on the deployment, reputation for about 90,000 sender identities and about 12 million IP addresses as of Feb 2009 have been computed. We note that email classification using RepuScore is 97.8% accurate.

Finally, this dissertation discusses future directions for Distributed RepuScore that allows organizations to maintain their personal reputation view to be shared among trusted peers. Distributed RepuScore enables a global reputation view while holding senders accountable at each organization instead of deploying it at a central authority.

ACKNOWLEDGEMENT

First and foremost, I would like to express gratitude to my dissertation director, Professor Brent ByungHoon Kang, whose insightful guidance has augmented my doctoral study. His constant help, suggestions, and willingness to meet me whenever I knocked at his office door deserve my further appreciation.

I am grateful to the dissertation committee members, Professor Anita Raja and Professor Xintao Wu, for their thoughtful suggestions that were extremely helpful in completion of this dissertation. I would also like to thank them for their warm advice on my career path.

I am would additionally like to thank Professor Yuliang Zheng and Professor Gail-Joon Ahn for their advice during my doctoral study. Further, I am indebted to members of the Infrastructure Systems Research Laboratory for their valuable feedback on my research.

I would to thank Ladar Levison and Ben Poliakoff for their contributions to my research by providing logs without which most of the experiments would not have been possible. I would like acknowledge Jeff Moss whose open source tool has been very valuable to this research.

Finally, but most importantly: my special thanks to my parents, brother and Vijita for their support, prayer, and love.

DEDICATION

To Guruji and my family
for everything.

TABLE OF CONTENTS

LIST OF FIGURES ix

LIST OF GRAPHS x

LIST OF TABLES xiii

LIST OF EQUATIONS xiv

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: BACKGROUND AND RELATED WORK 8

- 2.1 Email Classification based on content and Collaborative Blacklisting 10
- 2.2 Classification based on Sender's Identity 10
- 2.3 Classification based on Reputation/Certification Schemes 12
- 2.4 Reputation Management in Peer-to-Peer systems 14

CHAPTER 3: PRIVILEGE MESSAGING 16

- 3.1 P-Messaging: Components 18
- 3.2 Circle of Trust in P-Messaging 19
  - 3.2.1 Addition of a P-Server to the CoT 20
  - 3.2.2 Revocation of a P-Server from the CoT 20
  - 3.2.3 Advantages of CoT 21
- 3.3 P-Messaging Architecture 21
  - 3.3.1 Sender Architecture 22
  - 3.3.2 Receiver Architecture 23
- 3.4 Privileges in an email header: Privilege Tag 24
  - 3.4.1 P-Tag Creation and Maintenance 25
  - 3.4.2 Privilege-List Maintenance 25
- 3.5 Advantages of Privilege: A fine-granular and strongly-bound Identity 26
- 3.6 Privilege Messaging Implementation Details 26

CHAPTER 4: REPUSCORE FRAMEWORK 28

- 4.1 Sender Identity Techniques for Reputation Management 28

4.2 RepuScore: A Collaborative Reputation Framework 29

4.2.1 Design considerations for RepuScore 29

4.2.2 RepuScore Components 31

4.2.3 RepuScore Architecture 32

4.3 RepuScore Algorithm 35

4.4 Volume-Enhanced RepuScore Algorithm. 38

4.5 RepuScore Deployment 41

4.5.1 SpamAssassin Plug-in 42

CHAPTER 5: RESULTS 43

5.1 P-Messaging Evaluation 43

5.1.1 Signature Generation and Verification Time 43

5.1.2 Privilege Generation and Verification time 44

5.1.3 Effect of Cardinality on Collaborative Blacklists 44

5.2 RepuScore Evaluation 49

5.2.1 Format of Logs to RepuScore Algorithm 51

5.2.2 Experiments with Simulated Logs 52

5.2.2.1 Effect of α on trusted Sender 52

5.2.2.2 Initial Values for RepuCollector 54

5.2.2.3 Resilience to Sybil Attacks 55

5.2.3 Results from the Deployment 56

CHAPTER 6: CONTRIBUTIONS AND FUTURE WORK 72

6.1 Contributions 72

6.2 Distributed Reputation Management Framework 74

6.2.1 Distributed RepuScore Architecture and Algorithm 76

6.2.2 Effect of Synchronization 77

6.2.3 Evaluation of Distributed RepuScore 78

REFERENCES 80

LIST OF FIGURES

FIGURE 2.1: Comparison of Privilege Messaging with current technologies. 9

FIGURE 3.1 Granularity of Privilege Messaging Identity as compared to Domain-based solution and User-based White-lists. 17

FIGURE 3.2: Circle of Trust among the Privilege Servers. 20

FIGURE 3.3: P-Messaging Sender Architecture. 21

FIGURE 3.4: P-Messaging Receiver Architecture. 22

FIGURE 3.5: Microsoft Outlook Plug-in with new option: “Send with Privilege”. 27

FIGURE 3.6: Microsoft Plug-in that displaying list of Privileges. 27

FIGURE 3.7: Thunderbird plug-in with a new option: “Send with permissions”. 27

FIGURE 3.8: Thunderbird plug-in with a login prompt. 27

FIGURE 4.1: Hierarchical architecture for RepuScore with each RepuCollector receiving reputation information from multiple RepuServers. 32

FIGURE 4.2: The function e^{-x} function is monotonically decreasing function. 40

FIGURE 4.3: SpamAssassin plug-in collects statistics from mail servers. 41

FIGURE 4.4: Email classification using SpamAssassin plug-in. 41

FIGURE 6.1: Distributed RepuScore Architecture where RepuCollectors synchronize with friend-peers. 75

FIGURE 6.2: Effect of synchronization in distributed RepuScore 78

LIST OF GRAPHS

GRAPH 5.1: The overhead of P-tag using P-Messaging compared to PGP signed and unsigned emails. 44

GRAPH 5.2: The overhead of P-Tag verification compared to PGP signature verification. 44

GRAPH 5.3: Percentage of incorrectly classified email per day without receiver collaboration to maintain a blacklist. 45

GRAPH 5.4: Percentage of incorrectly classified email per day where receivers within an organization collaborated to maintain a blacklist. 46

GRAPH 5.5: Percentage of incorrectly classified email per day where all 5 organizations collaborate to maintain a global Blacklist. 47

GRAPH 5.6: The change in the reputation of a trusted domain that transmits spam after reputation interval 30 as a function of α. 52

GRAPH 5.7: In the modified RepuScore algorithm, a high value of α (other than 1.0) implies gradual increase, but fast decrease in reputation. 53

GRAPH 5.8: RepuScore's resilience towards Sybil attack. 54

GRAPH 5.9: Percentage of Authenticated Emails classified using RepuScore. 55

GRAPH 5.10: Around 10% of the authenticated sender identities were credible senders; while about 32% were known spammers. 56

GRAPH 5.11: Number of sender identities with lifetime of 1 day and 2 days plotted against their first appearance. 56

GRAPH 5.12: Cumulative distribution of sender identities as a function of reputation. 97.8% of the identities had reputation of 0 or 1. 57

GRAPH 5.13: The distribution of sender identities vs. their lifetime. The number of sender identities decreases as the lifetime increases. 58

GRAPH 5.14: Percentage of good (or bad) sender identities to total number of sender identities as plotted against lifetime. 59

GRAPH 5.15: Average reputation of all sender identities with the same lifetime. 59

GRAPH 5.16: Number of times a sender identity changed from good to bad or vice-versa. 60

GRAPH 5.17: Volume-Enhanced RepuScore reacts based on the email volume for a popular free email provider. 60

GRAPH 5.18: Volume of the Spam and Email noticed at receiver organization 1. 60

GRAPH 5.19: RepuScore: Reputation of the free email provider computed using information from two receiver organizations. 61

GRAPH 5.20: Volume-Enhanced RepuScore: Using volume, the global reputation is closer to the reputation from both the receiver organizations. 61

GRAPH 5.21: Reputation of IP address over the lifetime of the deployment. 62

GRAPH 5.22: The reputation of a well known free email provider computed over 170 days. 62

GRAPH 5.23: The number of IP addresses as a function of their lifetime. As lifetime increases, the number of domains decreases. 63

GRAPH 5.24: The number of authenticated sender identities as a function of lifetime. 63

GRAPH 5.25: The average reputation of the IP addresses as a function of the lifetime. 64

GRAPH 5.26: The average reputation of a sender identity as a function of lifetime. 64

GRAPH 5.27: The number of IP addresses with lifetime 0 and 1 that show up during the time of deployment. 65

GRAPH 5.28: The number of sender identities that appear with Lifetime 0, 1 that appear during the time of deployment. 65

GRAPH 5.29: The daily percentage of IP addresses that were accepted, rejected and with no reputation. 66

GRAPH 5.30: The percentage of rejected emails, accepted emails and with emails with no reputation computed by RepuScore. 66

GRAPH 5.31: The percentage of authenticated sender identities that were rejected, accepted and with no reputation. 67

GRAPH 5.32: The percentage of authenticated emails classified using RepuScore. 67

GRAPH 5.33: The percentage of False Positives plotted against number of days. 68

GRAPH 5.34: The percentage of False Negatives plotted against number of days. 68

GRAPH 5.35: The total amount of emails received and classified using RepuScore against number of days. 69

GRAPH 5.36: The percentage of emails classified using RepuScore. 69

LIST OF TABLES

TABLE 5.1: shows the average percentage of incorrectly classified spam. 48

TABLE 5.2: Mail Logs from Postfix Server. 50

TABLE 5.3: Sample logs from simulated RepuServer Table. 50

TABLE 5.4: RepuCollector and CA reputation tables. 50

TABLE 5.5 shows the values by distribution against the minimum good reputation. 57

LIST OF EQUATION

EQUATION 4.1: RepuServer algorithm computes sender’s reputation using good rate. 35

EQUATION 4.2: Modified RepuServer algorithm. 36

EQUATION 4.3: Local RepuCollector Reputation. 36

EQUATION 4.4: Central Authority computes global reputation. 37

EQUATION 4.5: Instantaneous value of α based on the volume of email. 39

EQUATION 6.1: Distributed RepuCollector while synchronizing with peers. 76

EQUATION 6.2: The threshold to pause synchronization with peers. 76

CHAPTER 1: INTRODUCTION

Email is a simple and cost effective messaging technology that has evolved into a universal mode of interaction among users. The day-to-day activities are contingent upon email infrastructure's consistent and reliable operation. Unfortunately, email architecture has not evolved proportionally as compared to Internet's sudden growth over the years. The original infrastructure was designed for communication among relatively few machines. The current email infrastructure is mostly based on the original design leading to a multitude of risks and threats faced by the users and their organizations. Unsolicited email has now reached epidemic proportions, severely limiting the email infrastructure's usability (Gomes 2004). Phishing (Milletary 2006), yet another threat, fools the recipients into divulging their financial information by redirecting them to a masquerading site. Spam presently contributes to about 95% of all email on the Internet and estimates place the financial losses due to phishing around $2.8 billion a year.

Recent legislations (such as CAN-SPAM (CAN-SPAM act) and other State Junk Email Legislations) that attempt to curtail unsolicited emails need to monitor an extremely large amount of email communications. To deal with the deluge of emails, organizations have adopted spam filters or blacklists that identify spam-propagating senders. These techniques provide an excellent mechanism to weed out most unsolicited emails; however, they could potentially blacklist a legitimate sender. The effectiveness of blacklists therefore is subjective to the deploying organization. Organizations usually consider a false positive[1] 10 times more seriously than a false negative[2]. Therefore, financial losses are not only due to the unsolicited bulk email but also from false positives (Finnegan 2006).

These observations motivate a fundamental need to enforce sender accountability over the email infrastructure. By holding sender accountable, receivers would be able to hold senders accountable for the emails they transmit. For example, consider the situation where the senders consistently spam, by holding senders accountable, the receivers can reject sender's emails. Though different techniques (such as blacklists) are available for email classification, sender accountability dictates a need to monitor the senders over long periods.

To monitor the senders for extended periods, an irrefutable identity is essential. For example, users on the Internet can send emails using IP address as the identity. Unfortunately, spam Botnets (Ramachandran 2006) also use IP address to proliferate unsolicited emails. A spoofed email[3] though contains an envelope IP address, the IP address cannot be held accountable for the actions that they have not performed. However, the IP address is accountable for the spoofed email.

In an effort to stop spoofing, senders started using sender identity techniques, discussed in Section 2.2, such as DKIM (Allman 2005, Peterson 2006), SPF (Wong 2004) and SenderID (Microsoft 2004). Currently, these techniques are a basis for determining the sender's history of adherence to best email practices (Jordon 2006).

[1] False Positive refers to the unsolicited emails being classified as solicited emails.

[2] False Negative refers to the solicited emails being classified as unsolicited emails.

[3] In Email Spoofing, an email's sender address and other parts of the header are altered such that it appears to have originated from a different source.

Presently, about 35% of all emails over the Internet use one of the sender authentication systems (Peterson 2006). The experiments conducted during this dissertation also demonstrate that about 33% of the incoming emails were authenticated after first filtering emails using blacklists.

The above techniques create a weakly-bound identity. Such an identity can be considered weakly-bound as senders create their own identity. To be able to classify emails from senders, there is a requirement to maintain a history of actions of these identities. In the case of a weakly-bound identity, the senders' identities can only be revoked/ corrected by the senders who create the identity. A strongly-bound identity, on the other hand, requires organizations to register their identity with a third party. Accredited DomainKeys (Goodrich 2005) extends the presently deployed sender identity techniques with the use of a central authority. Even with a strongly-bound identity, there is a need to observe sender behavior. A strongly-bound identity allows the central authority to revoke a spammer sender identity.

To establish sender accountability, this dissertation first discusses the design of the Privilege Messaging (P-Messaging) Framework that enables an authenticated sender to send emails while enabling the receivers to verify the email's privileges before creating content in the user's mailboxes. A privilege can be viewed as a credential associated with a group of users used to identify the senders.

Statistics collected with the help of a deployment at a single organization, as discussed in Section 5.2.3, show that adoption of a sender identity technique by an organization does not necessarily prove a sender's adherence to a good mail practice. A sender's identity alone does not allow receiver-organizations to differentiate between a credible sender identity and an unscrupulous one. It has been noted that spammers have been the early adopters of the sender identity schemes. Preliminary experiments discussed in Section 5.2.3 show that about 89% of the sender identities that use SPF and DKIM were classified as spammer.

Given sender identification schemes, a mechanism for the receivers to evaluate the sender's past adherence to good email practices can be designed. This dissertation describes RepuScore: a reputation framework which computes senders' reputation using receiver collaboration to enforce sender accountability.

Organizations such as Gmail (Taylor 2006) use reputation framework to classify emails. Email reputation has been demonstrated to be highly effective in email classification due to the ability to maintain the history of the sender's adherence. Due to large number of users, large organizations can maintain accurate reputation for large number of senders. Smaller organizations desire reputation information for a large number of senders, but cannot attain it themselves due to their limited operations. From our deployment, we noticed that during a single year of deployment, a small organization computed reputation of about 105,000 IP addresses where as an Email Service Provider collected reputation for 12 million IP addresses. Small organizations would benefit from sharing reputation about the senders they interact with. The reputation collected from multiple organizations allows receivers to access a relatively large history for a sender's past actions. Such a perspective allows a receiver organization to accept emails from senders about whom no prior information is available. With the help of receiver collaboration, a sender's spamming activity would be reported to all receivers. Such a mechanism places the onus on the senders to reduce the amount of unsolicited emails they send.

RepuScore provides organizations with the benefit of collaboration. RepuScore eases the overhead of reputation collection and computation with the help of distributed architecture. RepuScore supports a central architecture with minimal overhead (Jakobsson 2004) for organizations to enforce sender accountability.

RepuScore maintains a history of the spam rate as a measure to evaluate the senders using the Time Sliding Window Exponentially Weighted Moving Average (TSW-EWMA) algorithm proposed by Biswas et.al (2005). RepuScore is resistant to Sybil attacks (Srivatsa 2005, Yu 2003, Yu 2006) by valuing a reputable participant’s vote higher than that of a less reputable one. To thwart Sybil attacks, RepuScore employs the Weighted Moving Algorithm Continuous (WMC) (Yu 2002) algorithm.

In Chapter 5, this dissertation discusses the evaluation of the RepuScore framework using a) simulated logs; and b) with the deployment at three receiver organizations (from 10/9/2007 – to 04/26/2008). Using the simulated logs to demonstrate the features of RepuScore framework, the framework is shown to withstand Sybil attacks from colluding reputation reporters.

For the real-world deployment, a RepuScore plug-in for SpamAssassin has been designed which employs existing SpamAssassin plug-ins. The RepuScore plug-in verifies the sender identities and calculates senders’ reputation at a single organization. From the deployment, this dissertation discusses the results of the deployment where reputations for over 16,500 sender identities and 58,000 IP addresses have been computed. Further, this dissertation discusses:

a. Two variations of the RepuScore algorithm: a) using spam rate; and b) using both spam rate and email volume. As using only the spam-rate, variations in the received email volume significantly impacted the reputation of the sender, Volume-Enhanced RepuScore was designed as an extension that uses both spam-rate and email volume for computing reputation.

b. the design of a RepuScore SpamAssassin plug-in. The plug-in uses existing SpamAssassin plug-ins to verify the sender identity and transmits the information to a local RepuServer.

c. Observations and statistics from our deployment from RepuScore:

i) With knowledge of only 42% of the sender identities (11% legitimate, 32% spammers), RepuScore classified 72% of the authenticated emails.

ii) 97.8% of the sender identities had a reputation of either 0 or 1 where the reputation is in the open interval (0, 1).

iii) Sender identities with a low reputation score have a shorter average lifetime (17.47 days) as compared to ones with a high reputation score (61.9 days).

iv) A large percentage of sender identities send an email only on a single day.

v) Reputation can be accurate in determining if a sender identity is a spammer.

The global reputation is a single view that is used to enforce sender accountability. While a global reputation labels a specific sender as that of a low repute, another receiver organization might have a contradictory view. Maintenance of a single global view drowns out contradictory views resulting in directed attacks where spam senders could maintain high reputations by sending legitimate mail to reputable domains while sending unsolicited emails to others. It is imperative to maintain conflicting views rather than a single view that could be enforced by all senders.

Sharing reputation among different entities poses privacy issues. Organizations would wish to share reputation collected only with a selected trusted list of peers rather than the entire world. The design of RepuScore requires nodes to be time-synchronized to calculate reputation based on inputs from a majority of the receivers. Time-synchronization among multiple receiver organizations is difficult to achieve.

Due to these limitations, this dissertation proposes Distributed RepuScore as an extension of the RepuScore, where each receiver organization maintains its own reputation by eliminating the requirement for a central authority. To achieve higher accuracy rates, the receiver organizations synchronize the reputation they compute about a set of sender identities with each other. Distributed RepuScore could allow organizations to choose their peers to synchronize reputations with. For example, governmental organizations can participate in reputation sharing without the need to share data with civilians.

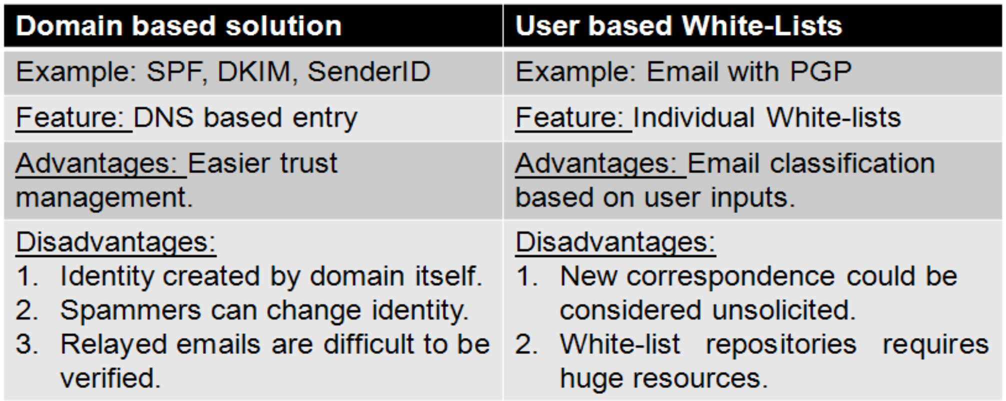
This dissertation is organized as follows. Chapter 2 discusses the background and related work including sender identity frameworks, reputation management frameworks and finally, available email reputation management techniques. Chapter 3 discusses a novel flexible cardinal identity framework. In Chapter 4, this dissertation discusses RepuScore framework, the first open source email reputation framework. Chapter 5 discusses the result of our experiments. In Chapter 6, this dissertation discusses the contributions and future work. Finally, in Chapter 7 concludes with the discussion of our main contributions.

CHAPTER 2: BACKGROUND AND RELATED WORK

Network-based email communication has existed since the early days of ARPANET (Hardy 1996) enabling a small, close-knit group to communicate electronically. Today, even with the extensive usage of email infrastructure (Gomes 2004), there is no mechanism to provide strict sender accountability. As senders are not accountable, there is no way to guard against fraudulent mailers sending unsolicited messages to others. To combat this, multiple filtering technologies have been developed that weed out most, but not all of the unsolicited email.

One of the mechanisms to address spoofing is to identify the senders. The sender identity techniques have been implemented using: a) User-based identity; and b) Domain-based identity. User-based identity systems, such as PGP, identify a specific user to the receivers. Each user creates an identity that is submitted to a central authority. The identities are used to identify the senders and classify emails from them.

Domain-based identity solutions authorize specific IP address to transmit emails. These IP addresses are identified in the DNS records by the respective organization's System Administrators. Domain-based identities such as SPF, DKIM and SenderID, discussed in Section 2.2, were developed to deal with email spoofing. Technologies can be considered as a weakly-connected – weakly because the identity is created by the organizations themselves. A recent study reports that 35% of all email is authenticated using one of the sender identity techniques (Ironport 2006). Organization can setup the sender identities themselves. Spammers have started adopting these solutions to create



Domain based solution	User based White-Lists
Example: SPF, DKIM, SenderID	Example: Email with PGP
Feature: DNS based entry	Feature: Individual White-lists
Advantages: Easier trust management.	Advantages: Email classification based on user inputs.
Disadvantages: 1. Identity created by domain itself. 2. Spammers can change identity. 3. Relayed emails are difficult to be verified.	Disadvantages: 1. New correspondence could be considered unsolicited. 2. White-list repositories requires huge resources.

Figure 2.1: Comparison of Privilege Messaging with current technologies.

their own identities to be able to propagate spam. Accredited DomainKeys allows a third party to maintain the identities. Such a mechanism allows the identities to be strongly-connected; an organization needs to notify the central authority before changing them.

Figure 2.1 shows the comparison between domain-based identity and user-based identity. When domain is used as an identity, a complete domain is either white-list or black-listed. For example, when a domain is blacklisted, honest-users of the domain would not be able to send emails. Therefore, organization should maintain granular identities smaller than that of the domain.

On the other hand, when using an email ID as an identity to check against both white-lists and black-lists requires huge recourses. A new correspondent might be considered as unsolicited unless the sender is listed in the white-list beforehand. These lists can become extremely large to incorporate multiple users' lists. The management of these lists can become extremely challenging: for example, white-listing need to be maintained by the users, the mail servers and the domains.

Email classification schemes can be classified as: a) the characteristics of the email content and blacklisting; b) the sender identity credentials; c) reputation/certification schemes.

2.1 Email Classification Based on Content and Collaborative Blacklisting

Word filters (Ahmed et.al 2004) search for patterns and remove the most obvious spam; however, spammers have often circumvented word filters by using misspelled words. Thus, word filters requires regular updates of the misspelled words used in unsolicited emails. Rule-Based scoring mechanisms check for keywords and use rules to analyze emails depending on the score received by a particular email. Bayesian filters (Sahami 1998) perform lexicographical and statistical analysis on the email for words and/or phrases depending on the recipient's previous spam emails.

Incorporating user feedback at the Message Transfer Agent (MTA) level forms the basis of collaborative filters (Gray 2004). With collaborative filters, an unsolicited email is filtered with the help of users' feedback on falsely classified emails. A combination of different techniques provides a reliable means to classify an email (Leiba 2004). SpamGuru (Segal 2004) employs multiple techniques such as word filters. Chung-Kwei algorithm (Rigoutsos 2004), a pattern discovery technique is also employed to classify emails.

Other techniques such as HoneySpam (Andreolini 2005) borrows the idea of Honeypots (Provos 2004); Social Network based classification and changing email infrastructure from push to pull architecture (Duan 2005, Neustaedter 2005) are suggested to identify spammers. However, these systems do not address the essential problem of unrestricted access to others' mailboxes.

2.2 Classification Based on Sender's Identity

Blacklisting IP addresses by users is comparatively simpler and computationally less intensive than other techniques. This process keeps a list of IP addresses identified as spammers and a white-list for legitimate users. Real-time Blackhole List (RBL) (Realtime Blackhole List 2002) works similar to Blacklist IP, but RBLs are not manually updated by individual organizations but by RBL operators who maintain the public RBLs.

PGP (Price 2003, Zimmerman 1995) allows verification of a sender's email address based on the sender's domain identity. PGP is an email granular service containing a list of individual users' public keys. User contact management based on PGP keys can provide the benefits of identifying trusted correspondents as well as verifying the email's integrity. However, utilizing PGP incurs the overhead as each individual user needs to maintain a white-list and a black-list. A new correspondence might be considered unsolicited unless the email id is enlisted beforehand. The size of the white-list can grow unbounded because the local and global white-lists may need to list all the legitimate email IDs on the Internet.

SenderID (Microsoft 2004) addresses the problem of spam and phishing by validating an email's origin, i.e., by verifying the IP address presented by the email against the sending domain's registered domain's email servers. SenderID is a domain-granular service validating a sender's domain.

Sender Policy Framework (SPF) (Wong 2004) is a technique that has been introduced to prevent email forgery. Senders identify the email server in their Domain Name Service (DNS) entries. The receivers validate the sender's email servers by evaluating the DNS records. The DNS records indicate the sender's adopted policy, for instance, the list of email servers allowed to send email by the domain. When an email is received, the receiver checks the sender’s policy specified through their DNS records. If the sender’s email server is not the one specified in the policy, the email is considered unsolicited.

Domain Key Identified Mail (DKIM) (Allman 2005), attempts to reduce the traffic on the network by enabling each sender to publish a public key through its DNS records. The email server signs each outgoing email. The receiver verifies the digital signature by retrieving the public key of the sender from their DNS records; thereby verifying the senders’ authenticity. However, the Public Key Infrastructure (PKI) key pair is generated by each domain itself. This allows spammers to adopt DKIM without being accountable. By using a Certificate Authority (CA) (Goodrich 2005), the senders can be made more accountable. DKIM presently does not enforce this restriction.

Accredited DomainKeys adds a central authority to the DomainKeys architecture (Goodrich 2005). The centralized authority, called the Accreditation Bureau, maintains the sender domain’s public key. Users conform to a specified usage policy and adherence to the policy is periodically checked. A history of the past adherence should be used to check the adherence to the specified usage policy.

2.3 Classification Based on Reputation/Certification Schemes

To create a group of senders whose prolonged history vouches for its email best practices, a reputation management system should use a domain name rather than the sender domain’s IP addresses. Basing reputation on the domain name strongly ties an organization with its past email activity because (i) an IP address does not intuitively translate to a domain name (Dewan 2004); (ii) multiple organizations can share an IP address; (iii) credible organizations in general would maintain their domain name for a longer period of time than their IP addresses.

SenderPath's SenderScore Certified (Sender Score Certified et.al), Habeas' Safelist (Habeas Safelist et.al) and Goodmail's Certified Email (Goodmail Systems et.al) are certification and accreditation services that allow bulk senders to obtain third party certification. These systems are not reputation systems: as the senders maintains the reputation and not the receivers. Spammers could adopt these systems and use them to transmit unsolicited content without being held accountable.

Return Path's Sender Score (Return Path et.al) and Habeas' SenderIndex provide reputation management per sender's IP address. SecureComputing's TrustedSource (CipherTrust 2006) provides a global reputation system that receives reports from deployed email servers in different organizations. Reputation based on IP addresses is not effective, as an IP address cannot be bonded to specific organizations (Dewan 2004). For instance, when multiple organizations share an IP address, spammers in a single domain can affect the reputation of users in other organizations. Moreover, if organizations move to another service provider, their past actions are no longer be attributed to them. Reputation should be more closely associated with the organization, utilizing the organization's domain name.

Project Lumos (Brondmo 2003) was proposed as an effort to provide reputation among collaborating ISPs. Receiver organizations provided feedback as to whether a sender was a spammer or otherwise. The algorithm was designed to compute reputation based on the activity of the previous 180 days. Project Lumos was designed to consider the weighted average of previous and present reputation of the senders. Project Lumos' proposal has been made public; but the project does not seem to be deployed. Moreover, Project Lumos' design does not consider Sybil attacks.

Google's reputation service (Taylor 2006) identifies the senders using best-guess SPF or DKIM and computes the sender's reputation based on user inputs. This system demonstrated a high accuracy in classifying bulk of Gmail's incoming emails. Google's reputation service computes reputation from a single domain. It does not provide a mechanism for collaborative reputation sharing among different entities. Taylor et.al suggests the need for a third party reputation framework, i.e., collaboration among multiple organizations to compute reputation.

Cloudmark's Network Classifier (Prakash 2005) is a community-based filter-system where multiple agents submit feedback about emails to nomination servers which require multiple users to confirm the claim that an email is spam. This information is submitted to a central server known as the Trust Evaluation System which computes a global view for an email's fingerprint. The Cloudmark paper advises not to use authenticated domain name as a fingerprint, as this would lead to a high multiplicity and cross-collision rate. RepuScore uses an authenticated domain name to maintain reputation for each sender instead of using a fingerprint for each email.

2.4 Reputation Management in Peer-to-Peer Systems

Reputation management techniques have been used in agent-based systems (Shmatikov 2003, Swamynathan 2008) as a mechanism to evaluate trust. In multi-agent systems, peers use reputation to evaluate other agents to select the best course of action to maximize their own outcome (Papaioannou 2004). Reputation systems have been prone to Sybil attacks (Douceur 2002) where a single attacker uses multiple identities to submit multiple reputation votes about its peers. Such attacks are detrimental to honest users and amicable to the attacker.

To protect against deception and attacks in cooperative reputation systems, inputs from honest users are considered more valuable than the inputs from dishonest users. The effectiveness of reputation protocols can be measured by their success in thwarting Sybil attacks. User-personalized reputation in addition to the global sender reputation is one mechanism to harden the reputation frameworks (Chirita 2004).

Using eigenvectors, EigenTrust (Kamvar 2003) uses global reputation to identify malicious peers. Future reputation is calculated based on the present normalized trust reputation of all the peers. Due to the feedback mechanism, EigenTrust is a self-policing system that regards a trusted peer more than that of a peer of low-repute. Such a mechanism is helpful in guarding against Sybil attacks.

Another reputation framework has been developed as an application-independent system (Yu 2002). This system considers the multi-player prisoners dilemma, where every agent tries to maximize its own profits while maintaining the trust of other nodes. The system also incorporates (Yu 2003) a mechanism to detect deceptions and reduce the effect of malicious votes from such peers.

Our framework builds architecture for email sender identity and email reputation along with an algorithm to compute reputation. In later sections, we discuss the architecture and elaborate the email algorithm framework.

CHAPTER 3: PRIVILEGE MESSAGING

To enable sender accountability on the email infrastructure, this chapter discusses sender identity. Sender identity techniques can be classified into two different categories: a) weakly-bound; and b) strongly-bound. A weakly-bound identity system allows organizations to create their own identity and maintain them. For example, SPF, DKIM, SenderID and DomainKeys are sender identity systems where organizations create their own identities, sometimes, in their DNS records. Each organization controls the identity it creates. These technologies have been developed as a mechanism to combat spoofing and not necessarily to be used as a sender identity. A strongly-bound identity requires a mechanism to bind an identity to an organization. Such a mechanism requires a third party to maintain and evaluate these identities. Accredited DomainKeys (Goodrich 2005) uses a central authority that controls the identities.

Apart from classifying sender identities as weakly-bound and strongly-bound identities, the sender identities are domain-based and user-based. We note that a user-based solution requires extensive resources, both storage and computation, to maintain a history of accountability, but would be useful in maintaining an accurate sender accountability. In contrast, we note that a domain-based solution entails lesser storage and computation for each identity; however, it averages the information for all users at a single organization. Therefore, a domain-based solution would not be as accurate as the email id. Finally, we note that holding a domain accountable for their users' actions

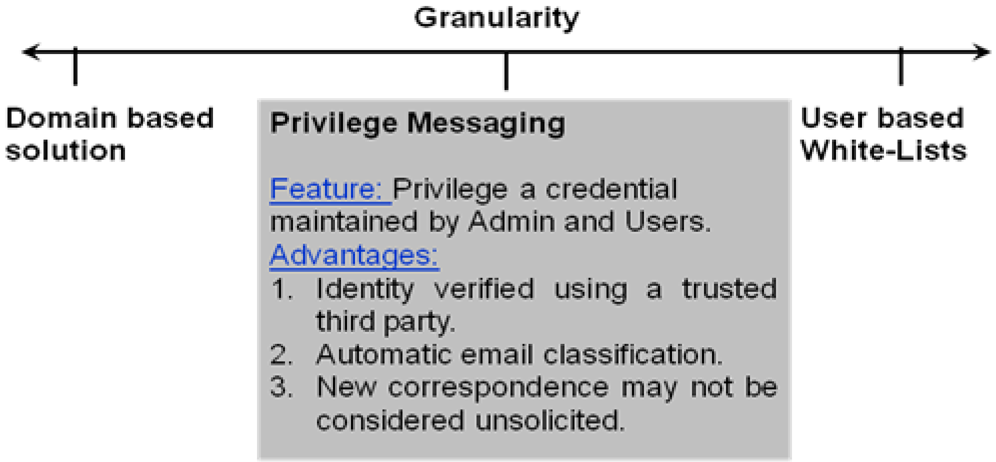

Granularity
Domain based solution
Privilege Messaging
Feature: Privilege a credential maintained by Admin and Users.
Advantages:
1. Identity verified using a trusted third party.
2. Automatic email classification.
3. New correspondence may not be considered unsolicited.
User based White-Lists

Figure 3.1 Granularity of Privilege Messaging Identity as compared to Domain-based solution and User-based white-lists.

enables peer organizations to force the said sender domain to revoke a user. However, such a mechanism is not available when sender accountability is to be maintained for each email address.

This dissertation discusses the design of Privilege Messaging (P-Messaging), a fine-granular1 sender identification technique. P-Messaging creates identities that can be customizable allowing different users to be added or removed from it. The identity is maintained independently from email infrastructure.

A *privilege* is a fine-granular identity associated with a PKI key pair and a group of users. P-Messaging stipulates that an email can be verified based on the *privileges* it holds. The administrators set the granularity of a privilege. Each email id can be assigned multiple privileges. For instance, an email ID can be associated with a department in a school and another for project team for a class.

P-Messaging creates and maintains a Circle of Trust (CoT) among P-Messaging

[1] A granular identity allows the email administrators to modify the number of users to the identity.

Servers to identify trusted senders. A CoT is a trust relationship among P-Messaging Servers due to the mutual trust they place on a third party: Certificate Authority (CA). Using CoT, qualified trust can be placed on a sender by a member. Each of the P-Messaging Server in turn places a high level of trust on each of their privileges.

The following section discusses the components of P-Messaging, architecture for CoT, followed by the architectures of P-Messaging.

3.1 P-Messaging: Components

This subsection discusses the components of P-Messaging:

1. P-Messaging Server
2. P-Messaging Privilege Verifier
3. P-Messaging Trust Authority

P-Messaging Server

Architecturally, P-Messaging Server (P-Server) is a component between the user and the organization's Mail Transfer Agent. To send an email, the users at an organization interact with its P-Server. The P-Server validates the user and attaches the requested credentials (i.e., privileges; for example: UNCC_MEMBER, ISR_MEMBER, UNCC_STUDENT, UNCC_COIT, CLASS_ITCS6162 and CLASS_ITCS6112) to the email. The privileges-attached email interacts with the MTA to transmit the email. The P-Server provides the following services:

- User Authentication: authenticates users using challenge-response authentication (Wikipedia). With the help of a user authentication scheme, a P-Server associates privileges to be associated only with authenticated users.
- Privilege Lookup: lists all the privileges a user can send email with. Once the

users are authenticated, the Privilege Lookup service identifies the privileges that a specific user can send email with.

- Message Signing: attaches the privileges to an email using a digital signature. The signature uniquely-identifies the selected privilege.
- Privilege Administration: is an administrative interface to manage users and privileges. The administrators can add users to a privilege, create privileges etc.

P-Messaging Privilege Verifier

P-Messaging Privilege Verifier (P-Verifier) verifies the received privilege-emails from senders based on the attached signature. The P-Verifier provides two services:

- Message Authorization: verifies an email's digital signature to verify the privileges and checks if the receiver accepts the email.
- Privilege-list Maintenance: is an interface to maintain a list of privileges a user accepts. Only the privileges listed by the user are accepted; other privileges even though authorized are not accepted.

P-Messaging Trust Authority

The P-Messaging Trust Authority is the entity that creates the Circle of Trust among P-Servers by providing a certificate to each P-Server. With the help of a digital signature, the receivers can identify a valid P-Server. The Trust Authority maintains the history of adherence to the good email practices enforced by it.

3.2 Circle of Trust in P-Messaging

P-Messaging provides the capability to identify a valid P-Server before accepting any emails from them with the help of Circle of Trust (CoT). A Circle of Trust creates a group of senders and receivers who can place some qualified trust on each other. P-

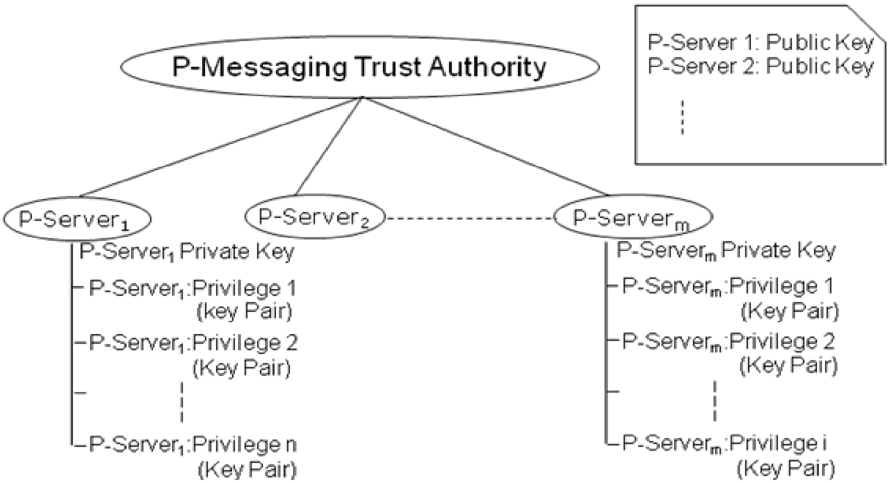

P-Messaging Trust Authority
P-Server 1: Public Key
P-Server 2: Public Key
P-Server$_1$
P-Server$_2$
P-Server$_m$
P-Server$_1$ Private Key
P-Server$_1$:Privilege 1 (key Pair)
P-Server$_1$:Privilege 2 (Key Pair)
P-Server$_1$:Privilege n (Key Pair)
P-Server$_m$ Private Key
P-Server$_m$:Privilege 1 (Key Pair)
P-Server$_m$:Privilege 2 (Key Pair)
P-Server$_m$:Privilege i (Key Pair)

Figure 3.2: Circle of Trust among the Privilege Servers using P-Messaging Trust Authority that allows a Privilege Server to be verified by other Privilege Servers.

Messaging framework can be deployed with mechanisms that determine a peer's trust. P-Messaging's CoT uses a peer's trust to enforce sender accountability. Honoring a P-Server's privileges, i.e., accepting email across domains is dependent upon the qualified trust that can be placed on the sender P-Server. If any P-Server sends unsolicited emails, the amount of trust placed on it should decrease. This section discusses how CoT is created and maintained.

3.2.1 Addition of a P-Server to the CoT

Figure 3.2 describes P-Messaging's hierarchical architecture, where the P-Messaging Trust Authority functions as a CA. All P-Servers trust the trust authority. Each P-Server must receive a certificate from the P-Messaging Trust Authority that can be used to verify the P-Server by other servers.

3.2.2 Revocation of a P-Server from the CoT

The P-Messaging Trust Authority can revoke a P-Server based on a prior agreement. One condition could be that a member of the sender's adherence is mandated

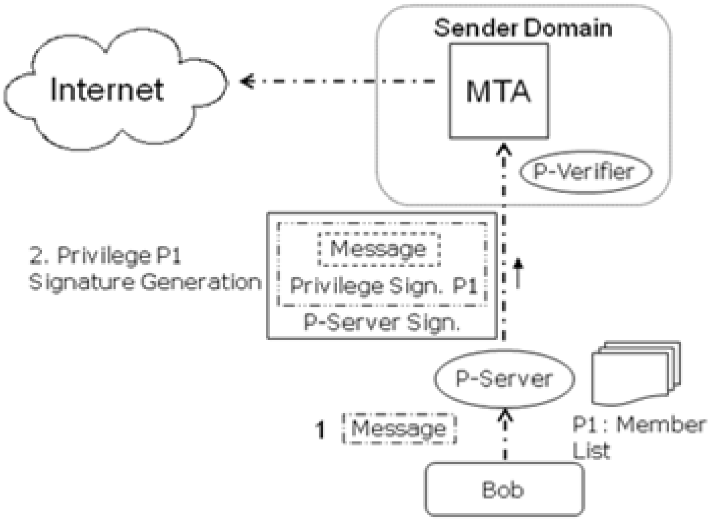


Figure 3.3: P-Messaging Sender Architecture: After the sender Bob is verified, the P-Server signs the email with a privilege specified in the Member List. The email is then sent from the P-Server to the MTA that relays the email.

to follow a common policy; if the sender does not conform to it, the sender could be revoked. Upon being revoked, a previously participating P-Server can request new certificates but under strict adherence to the common policy. Therefore, if a P-Server does not contain a privilege’s negative characteristics, the P-Server would be revoked from CoT.

3.2.3 Advantages of CoT

With the help of CoT, a P-Server can place qualified trust on other peer P-Servers. Honoring a sender’s privilege by receivers is based on the trust placed on the P-Server by the P-Messaging Trust Authority. With the help of this mechanism, a distributed identity system can be provided among different P-Servers with each P-Server is capable of creating its own privileges.

3.3 P-Messaging Architecture

P-Messaging architecture has two major objectives: a) sender architecture; and b)

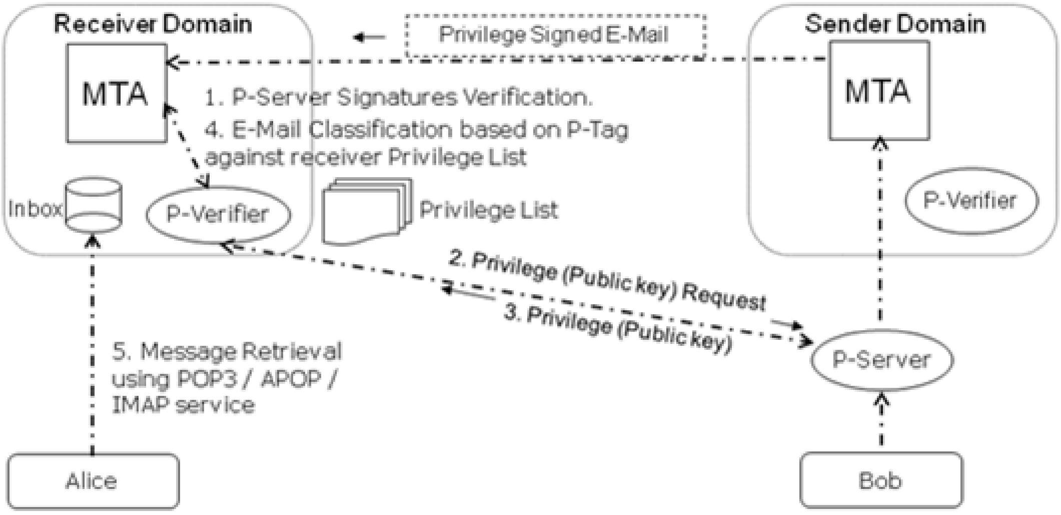

Receiver Domain
Privilege Signed E-Mail
Sender Domain
MTA
1. P-Server Signatures Verification.
4. E-Mail Classification based on P-Tag against receiver Privilege List
MTA
Inbox
P-Verifier
Privilege List
P-Verifier
2. Privilege (Public key) Request
3. Privilege (Public key)
P-Server
5. Message Retrieval using POP3 / APOP / IMAP service
Alice
Bob

Figure 3.4: P-Messaging Receiver Architecture: Once an email is received, the MTA passes the mail to the P-Verifier looks-up the public key of the privilege to verify the mail. Once the mail is verified, emails are classified according to the privileges specified in the Privilege List.

receiver architecture. A receiver P-Server, after sender identification, interacts with the receiver’s MTA before the email is delivered to the mailbox of the intended users based on the privileges the accepted by the users at the receiver organization. When sending an email, a P-Server interacts with the MTA to send the email with the sender organization’s user’s privileges.

3.3.1 Sender Architecture

A P-Server is a sandbox the MTA as shown in Figure 3.3. When sending an email, the sender P-Server first verifies the user, for instance, Bob. After user verification, the P-Server signs the email for the requested privileges. Bob select privileges, or the P-Server based on a simple rule-based engine from the Member List for every user at the P-Server. A P-Server signs an email and relays it through the MTA.

A receiver P-Server first verifies the sender P-Server; based on the trust on the P-Server verifies the privilege attached with the email. For example, when a P-Server installed at a university, the P-Messaging Trust Authority creates a key pair, for the P-Server, which is securely transmitted to the P-Server. The university's P-Server can then create multiple privileges, for example, for faculty called "faculty" and for students called "student".

The receiver accepts a privilege message only if it honors the sender's privilege. Without this, the message that is sent cannot be classified into privilege classes but into the underprivileged class. These classes are described in later sections.

3.3.2 Receiver Architecture

The receiver architecture is shown in detail in Figure 3.4. At the receiver's domain, upon receiving the email, the MTA verifies the privileges associated with the email with the help of the P-Verifier. For verifying an email, the P-Verifier evaluates the P-Server using the signature. Once the P-Server is verified, the privilege's public key is retrieved from the P-Server. Using the public key of the privilege, the privilege signed email is verified.

To retrieve the message, as shown in Figure 3.4, the client, for instance Alice, connects to the mail server to retrieve the messages. Using the additional header information, any email client can display the information in any desired format. The mail clients can show the different 'inboxes', where each inbox caters to a different class. In this way, the classification of the email into different classes provides users with the ability to view the messages according to the privileges accepted by them. This allows a faster lookup by classifying the emails at one location thereby, providing Quality of Service (QoS). For the example listed in Section 3.3.1, the receiver first verifies the university's P-Server and then verifies the privileges associated with it, by retrieving the PKI key pair for the privilege from the universities' privilege.

Once the mail is verified, the next step is to place the mail into classes. Based on user's Privilege List, P-Messaging defines three privilege classes:

Privileged Class

Privileged Class is a set of privilege-"inboxes" that hold emails that have been successfully verified and honored privileges by the receiver.

Underprivileged Class

Underprivileged class is a set of privilege-"inboxes" that are successfully verified, but not honored by the receiver. If the privileges presented by an email are deemed important, the receiver needs to subscribe to the privilege making it a privileged class rather than an underprivileged class.

No-privilege Class

The No-privilege classes form the lowest class where unsigned or emails whose authenticity cannot be ascertained are placed.

3.4 Privileges in an Email Header: Privilege Tag

The credentials attached to a privileged-email are referred to as Privilege Tag (P-Tag). The P-Tag has a digital signature evaluating the authenticity of the email's origin. P-Tag is designed to be extensible by allowing each P-Server to create its own privileges. Each P-Server acts a CA for the privileges it holds. The P-Tag format that is attached to the email's header as follows:

[P-Server]:[Privilege]

A privilege email is of the following structure:

{[Email] Privilege Signature}: {[Privilege Signature] P-Server signature}

A privilege is a double digital signature for the email, first using the privilege and then using the P-Server's private key by the P-Server. As discussed earlier, to verify a privileged email: first, the P-Server signature and then the privilege signature is verified.

3.4.1 P-Tag Creation and Maintenance

As part of Privilege management, apart from creation and maintenance of the privileges, the administrator of a privilege performs the tasks of adding and deleting/revocation of users. The privilege-administrator is responsible for:

1. Adding a Privilege to a user.
2. Deletion or Revocation of a user's Privilege.

Addition and revocation of privileges modifies a user's Member List. The Member List contains the privileges a user is authorized to send with. Users are added or revoked by the privilege administrator. If a user abuses the privilege, the administrator revokes the privilege. For a user's revocation, the member list is updated by removing the user from the Member List. As the privilege's private key is never revealed, the privilege administrator need not create another PKI key pair.

3.4.2 Privilege-List Maintenance

Each user at a receiver's organization maintains the list of accepted privileges by updating the Privilege List at the P-Verifier. The mail service provider assigns a default list of privileges. In the absence of a user's input, which could be quite common, the service provider's default privilege from the Privilege List will be used. In the future, some user-profiling or personalization could determine a privilege list on user's behalf.

3.5 Advantages of Privilege: A fine-granular and strongly-bound Identity

As shown in Figure 3.1, a privilege identity is a customizable identity that is more finely-granular than domain-based and more coarsely-granular than the user-based solutions. A privilege can be either a single user or a group of users that could be larger than the users at a single organization.

A domain identity has a single credential for the entire domain. In comparison, when P-Server is registered, the domain is given an authorization to manage their privileges, not the right to send emails. P-Messaging is installed over multiple P-Servers on a domain where each P-Server maintains multiple privileges.

DKIM allows public keys to be created and embedded into the DNS records, whereas P-Messaging requires the P-Server to publish its public key with a third party, the P-Messaging Trust Authority. Therefore, in P-Messaging, the message is accepted after verification of the sender by a trusted third party. P-Messaging’s identity is strongly-bound to the domain.

The credentials for the user-based white-lists are the individual email IDs as compared to privileges in P-Messaging. With white-lists, a new correspondent might be classified as an unsolicited sender. The benefit of P-Messaging is that a new correspondent may not be classified as an unsolicited sender. The variable-granularity of an identity allows the system to provide higher accuracy comparable to the email identity without significant investment of resources.

3.6 Privilege Messaging Implementation Details

This section first discusses the client software developed for P-Messaging using Outlook 2003 and Thunderbird.

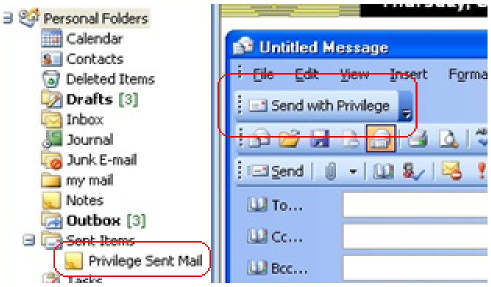
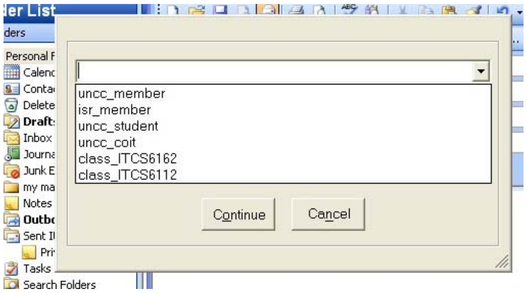
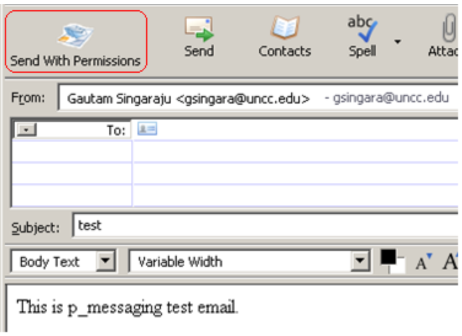


Figure 3.5: Microsoft Outlook Plug-in with new option: "Send with Privilege". A Privilege Sent Mail in the left pane that allows users to access all the privilege emails sent.



uncc_member
isr_member
uncc_student
uncc_coit
class_ITCS6162
class_ITCS6112

Figure 3.6: Microsoft Plug-in that displaying list of Privileges the authorized user can send an email with.



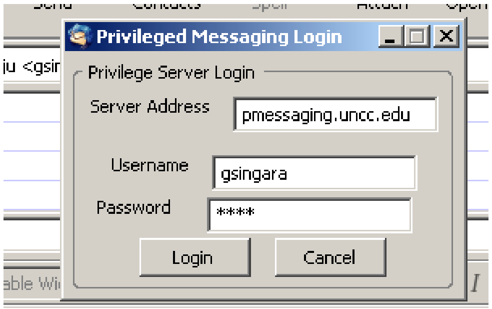
From: Gautam Singaraju <gsingara@uncc.edu> - gsingara@uncc.edu

To:

Subject: test

This is p_messaging test email.

Figure 3.7: Thunderbird plug-in with a new option: "Send with permissions" to send privilege-signed emails.



Server Address: pmessaging.uncc.edu

Username: gsingara

Password: ****

Figure 3.8: Thunderbird plug-in with a login prompt for the P-Messaging server.

To send and classify privilege emails, plug-ins for Outlook 2003 and Thunderbird has been designed. Figure 3.5 shows the Outlook plug-in with new options: "Send with Privilege" and with a new option in the left pane to show the list of all privilege emails by the sender. Figure 3.6 shows the list of privileges: UNCC_MEMBER, ISR_MEMBER, etc.

Figure 3.7 shows the Mozilla Thunderbird plug-in with a new option: "Send with Permissions" to send privilege-authenticated email. Figure 3.8 shows the Thunderbird's interface to login to the P-Server.

CHAPTER 4: REPUSCORE FRAMEWORK

A sender identity technique along with a reputation framework creates a trusted group of senders. A verified identity (through any existing authentication mechanism) is a required basis for maintaining sender's reputation. However, knowledge of the sender does not imply that the sender adheres to the best email practices. To enable receivers to identify the senders who adhere to the best email practices, the past history of the senders should be maintained. Reputation management framework maintains the history of adherence to a common policy about the best practices.

4.1 Sender Identity Techniques for Reputation Management

Systems designed using email id as a sender identity, such as PGP, when used to maintain reputation entails a huge overhead for vote collection, storage and computation. Instead of using email ids as identities, domain authentication schemes (DKIM, SPF or P-Messaging) could be used, thereby decreasing the number of identities needed. Such a mechanism is more scalable than the email id based reputation system.

About 35% of all authenticated email over the Internet is authenticated using SPF, DKIM or SenderID. A reputation framework evaluates the senders who are authenticated using these mechanisms. Such a mechanism will help evaluate the domains that adhere to a common guideline.

The lack of a centralized authority (Jakobsson 2006) has been noted as a main reason for the inability to tie email forgery to a single user or the organization. A central authority can maintain a group of reputable senders where each sender maintains a high reputation. Such a mechanism allows a common best email practice to be enforced among senders.

P-Messaging's CoT is designed with a central authority that can be used for reputation framework. The P-Messaging Trust Authority can use the reputation framework to maintain a group of trusted P-Servers. The reputation framework is used to work with current sender authentication schemes along with P-Messaging; and therefore, a different nomenclature is used.

4.2 RepuScore: A Collaborative Reputation Framework

RepuScore is a collaborative email reputation framework that allows participating organizations to establish sender accountability based on the senders' past actions. For a non-changing identity, RepuScore employs a) existing domain based schemes such as SPF, SenderID and DKIM; or b) their IP addresses. For the long-standing history, participating organizations locally collect information about the senders - from users or existing spam classification mechanisms that are submitted to a central RepuScore authority to compute a global reputation summary that is used to enforce sender accountability.

Multiple RepuScore algorithms have been designed that compute reputation gathered from organizations based on the sender's a) spam rate (RepuScore) or b) spam rate and email volume (Volume-Enhanced RepuScore).

4.2.1 Design Considerations for RepuScore

RepuScore should ease the overhead of vote collection and computation with the help of a distributed architecture. As each receiver-organization receives a lot of emails, a centralized vote collection incurs an excessive overhead. Such architecture allows each organization to collect votes from its users. However, distributing the reputation management creates additional challenges.

Since RepuScore employs a distributed reputation framework, it is susceptible to Sybil attacks (Srivatsa 2005, Yu 2006). In Sybil attacks, a malicious receiver manipulates the rating mechanism by creating multiple identities (also called Sybils) to give a) a higher rating to colluding senders; and b) a lower rating to legitimate senders. RepuScore should be able to thwart Sybil attacks by valuing a reputable participant's input more than that of a less reputable one.

The lack of centralized enforcement has been cited as the main obstacle in connecting email fraud to a particular user or organization. RepuScore should enable a centralized design with minimal overhead to create a trusted group of senders.

A reputation framework should facilitate in the creation of a group rather than just maintaining a group of blacklisted senders. RepuScore differs from other approaches because of the collaborative reputation based on the scores suggested by peers.

A reputation framework should accept a single reputation vote from each deployed organization. A large global organization might have multiple mail servers (or P-Servers), each situated in different geographic locations. If a reputation management framework considers votes from mail servers, an organization with a large number of mail servers would have greater say than those organizations that host a single mail server. Therefore, each organization should be allowed to vote only once considering all the mail servers in the domain.

The sender's reputation should decrease for bad behavior and increase in the absence of bad behavior; i.e., when spam is reported, the sender's reputation should decrease; else the reputation should increase.

The initial reputation for senders should be appropriately set, as an improper initial reputation would give high spam propagating domains an unfair advantage as their reputation would stay for longer intervals. In contrast, a low initial reputation would be unfair to a new domain, as peers would not accept its emails.

Reputation Aggregation Interval is defined as the sender's reputation should be computed every specific interval of time after receiver collaboration. A sender should adhere to good practices for a significant number of intervals to be considered good. Such a mechanism would make spamming unviable for a spammer as it would require a significant investment of resources, including both time and money. In addition, a quick reduction in reputation for non-adherence to the policy removes spammers from the trusted group of senders.

4.2.2 RepuScore Components

RepuScore framework has three components: a) RepuServer; b) RepuCollector; and c) Central Authority. A RepuServer as a mail server with the additional capability of verifying the users and collecting the reputation votes from them. Each local RepuServer collects votes from its users and email filters, aggregates the votes locally, and forwards them to the organization's RepuCollector.

A RepuCollector is an organizational-level service that aggregates votes from the local RepuServers and submits a single vote in a global reputation of peer RepuCollectors. As all the RepuServers belong to the organization, the RepuCollector belonging to the organization assumes that the RepuServers do not compromise the reputation.

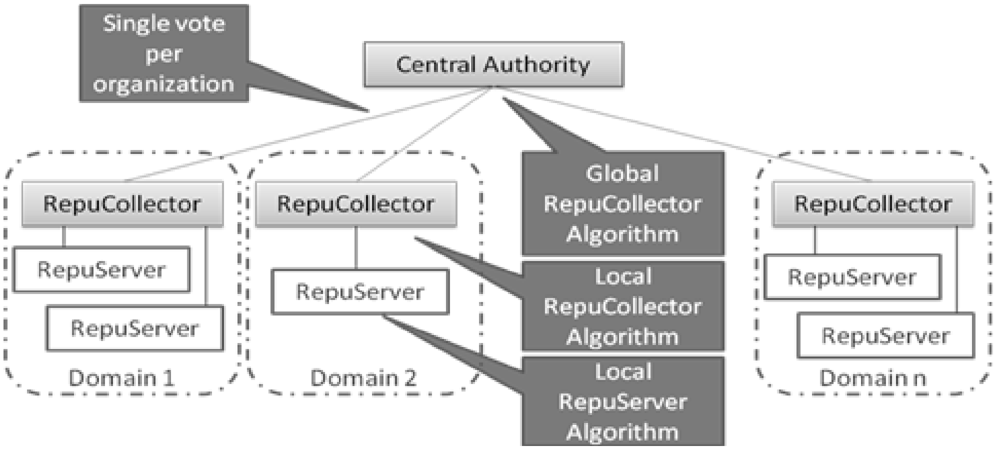


Figure 4.1: Hierarchical architecture for RepuScore with each RepuCollector receiving reputation information from multiple RepuServers. The RepuCollector transmits the reputation to a Central Authority.

The central authority computes the global reputation by computing reputation from multiple RepuCollectors. When Sybil attacks (Douceur 2002), where a single attacker takes multiple identities to thwart a reputation system by submitting incorrect information occurs, the algorithm should be able to secure against these attacks.

4.2.3 RepuScore Architecture

Figure 4.1 shows RepuScore's hierarchical architecture that is designed to ease reputation collection and computation as the number of participating domains increase. The RepuScore framework computes reputation based on the votes collected by each RepuServer. While collecting reputation votes, a RepuServer checks the validity of the reporting users. The user's votes are based on the evaluation of the sender's adherence to best practices. There are three major steps in RepuScore:

a) Reputation Vote Collection

b) Reputation Computation

c) Reputation Lookup Service

Reputation Vote Collection

As spam is subjective, another user might not consider an email regarded as spam by one user. Therefore, a global blacklist or white list would not be ideal as it would fail to represent the conflicting views of multiple users. RepuScore employs a social rating mechanism to consider the conflicting views of the users.

The receiver's RepuServer can maintain the number of emails received and the emails marked as spam for each sender RepuServer. The vote collection mechanism requires minimal participation from the users. For example, RepuServer collects the users' votes based on the users' implicit inputs; users only mark an incorrectly filtered-email as non-spam or to report a spam email that was not correctly filtered by the spam classifiers. (Many email services provide similar mechanisms for their users to report a spam email or an incorrectly filtered email.) Before accepting votes from the users, the RepuServer should validate the users.

The spam classifiers are also used along with users' input in collecting votes. A negative vote for a sender is entered when the spam filters determines an email as spam. Likewise, a positive vote for a sender is automatically made when the sender's email is not considered spam. In the event that the spam filter marks a legitimate email as spam, the users can mark the email as non-spam, submitting a positive vote for a sender to the RepuServer.

Reputation Computation

Based on the number of spam and emails collected, each RepuServer calculates the reputation of the sender RepuServer. RepuServer Reputation is defined as the weighted average of the reputation in the previous and the present intervals.

RepuScore calculates the reputation of a RepuCollector based on the reputation of the RepuServers maintained by the RepuCollector. A RepuCollector Reputation is defined as the aggregate reputation of the RepuServers in their domain in the present interval. Each RepuCollector calculates the local reputation for each peer RepuCollector. The computed reputation is digitally signed by each RepuCollector to maintain the integrity of the data. To provide a global perspective, the Central Authority should collect the locally computed RepuCollector's reputations.

RepuScore introduces a central authority that collects reputation votes from all the RepuCollectors and computes the global reputation for all RepuCollectors. The central authority verifies the RepuCollector's votes based on the digital signature. The central authority should make sure that the reputation collection is conducted once every interval. The central authority calculates a global reputation for each RepuCollector based on the change in its reputation as reported by peer RepuCollectors. The central authority takes into account the reputation of the RepuCollectors to compute the global reputation of the peer RepuCollectors. If the reporting RepuServers' reputation is below the participation threshold, their reputation votes are not factored into the global reputation.

Reputation Lookup Service

A reputation lookup service can be provided with the help of a third party lookup service. The reputation lookup service can be similar to Real-time Black Lists. Such a reputation lookup service can also provide a mechanism for the receivers to lookup the reputation of a sender's RepuCollector as reported by peers. An alternate way for receivers to determine reputation is by associating the reputation with a sender identity that can be vouched for by a third party. For example, in the case of Accredited

$Given\ a\ sender\ identity\ s, that\ has\ sent\ emails\ in\ interval\ m$

$$O_m = \frac{n(GE_m)}{n(E_m)} \qquad \dots\dots\dots\dots\dots [Equation\ 4.1a]$$

$$R_m = \propto \times R_{m-1} + (1 - \alpha)\ \times O_m \dots\dots\dots\dots\dots [Equation\ 4.1b]$$

$Where\ O_m\ is\ the\ Observed\ Reputation\ in\ Interval\ m$
$R_m\ is\ the\ Reported\ Reputation\ at\ Interval\ m - 1$
$n(GE_m)\ represents\ the\ number\ of\ good\ emails\ received\ in\ interval\ m$
$n(E_m)\ \ represents\ the\ total\ number\ of\ emails\ received\ in\ interval\ m$
$\alpha\ \epsilon\ (0,1)\ is\ the\ corelation\ factor$

Equation 4.1: RepuServer algorithm computes sender's reputation using good rate.

DomainKeys, the reputation can be embedded as the part of the seal that is supplied to the MTAs. When the client checks the DNS entries, the seal can be verified for the reputation.

In P-Messaging, the reputation will be used by the Trust Authority to calculate the reputation of the P-Servers. If the reputation of a specific P-Server is below the threshold, the P-Server will be removed from the framework.

4.3 RepuScore Algorithm

RepuScore computes the reputation at the individual P-Servers, at the organization and at the Central Authority. The algorithm is described below:

RepuServer Reputation Calculation

Peer RepuServers calculates a RepuServer's reputation. The reputation in RepuScore is always in the open interval (0, 1). A score of 1 indicates a highly reputable sender whereas a score of 0 indicates a sender with a low reputation. For all sender RepuServers, each receiving RepuServer maintains the number of emails received and the number among those marked as spam. Equation 4.1(a) demonstrates the algorithm to compute reputation of a RepuServer; the observed reputation is computed as the number

$Given\ a\ sender\ identity\ s, that\ has\ sent\ emails\ in\ interval\ m$

```
$if(O_m \geq R_{m-1})$
    $R_m = \propto \times R_{m-1} + (1-\alpha)\ \times O_m$
$else$
    $R_m = (1-\propto) \times R_{m-1} + \alpha \times O_m$
$endif$                ............... $[Equation\ 4.2]$
```

$Where\ O_m\ is\ the\ Observed\ Reputation\ in\ Interval\ m$

$R_m\ is\ the\ Reported\ Reputation\ at\ Interval\ m-1$

$n(GE_m)\ represents\ the\ number\ of\ good\ emails\ received\ in\ interval\ m$

$n(E_m)\ \ represents\ the\ total\ number\ of\ emails\ received\ in\ interval\ m$

$\alpha\ \epsilon\ (0,1)\ is\ the\ corelation\ factor$

Equation 4.2: Modified RepuServer algorithm maintains computes reputation.

$Given\ an\ identity\ l\ with\ series\ of\ Reported\ reputations\ (R_1, R_2, \dots, R_n)$
$from\ RepuServers\ (S_1, S_2, \dots, S_n)\ \ associated\ with\ a\ RepuCollector$

$$C_m = \left(\frac{1}{n}\right)\sum_{i=0}^{n} R_i \qquad \dots\dots\dots\dots\dots \text{[Equation 4.3]}$$

$Where\ C_m is\ the\ reputation\ for\ any\ sender\ identity\ in\ interval\ m$

Equation 4.3: Local RepuCollector Reputation calculates the average reputation of all its RepuServers

of good emails over the number of emails sent by a sender in a particular interval. Equation 4.1(b) shows the reported reputation calculated using modified time sliding window exponentially weighted moving average (TSW-EWMA) algorithm (Biswas 2005). The sender's reputation is based on the reputation in the previous interval and the observed reputation in the present interval. Correlation factor α indicates the amount of previous reputation considered for computation of the RepuServer's reputation in the new interval. If the correlation factor is high, the reputation of a sender takes a long time to increase or decrease, as a lot of weight is given to the previous reputation. However, if the correlation factor is low, the reputation increases or decreases very quickly since current actions are given additional weight. The effect of the correlation factor on

Given a sender identity s whose reputation is reported by Collectors with reputation $(C_1, C_2, \dots, C_n)$ *each each having Reputations* $(GR_1, GR_2, \dots GR_n)$*in interval* $m - 1$

$$G_s = \frac{\sum_{i=0}^{n}(C_i \times GR_i)}{\sum_{i=0}^{n} GR_i} \qquad \dots\dots\dots\dots\dots \text{[Equation 4.4a]}$$

```
$if\ (s \in set\ of\ RepuCollectors)$
    $GR_s = G_s$
$endif$                              ……………[Equation 4.4b]
```

Where G_r *is the global reputation of sender s in the interval m*

Equation 4.4: Central Authority computes global reputation. To thwart Sybil Attacks, different weight is applied based on the RepuCollector's reputation.

reputation in the experiments is demonstrated.

The reputation should increase slowly to check a long history of adherence, while the reputation should decrease quickly when the domain starts spamming. However, using Equation 4.1, the reputation either increases and decreases slowly or increases and decreases quickly. The Equation 4.2 should be modified to allow slow increase and quick decrease in reputation.

RepuCollector Reputation Calculation

The change in RepuServers' reputation is used to update the RepuCollector's reputation. Equation 4.3 shows the local reputation computation at a RepuCollector. The RepuCollector reputation is the average reputation of all of the reports from its RepuServers. This information is transmitted to a Central Authority that computes the global reputation. Such a mechanism can be shown to help in the creation of a trusted group of reputable senders.

CA Reputation Calculation

The central authority calculates the global reputation of the RepuCollectors based on a modified Weighted Majority Algorithm (WMA) called WMA Continuous (WMC) proposed by Yu et al (Yu 2002). The WMC algorithm has been used in peer-to-peer systems to detect deception.

Equation 4.4 demonstrates the Global RepuCollector reputation as the reputation-weighted average of the local RepuCollector reputation computed by each peer. The new reputation is computed once every interval and is valid for one interval.

4.4. Volume-Enhanced RepuScore Algorithm

An interesting observation from the deployment was that the reputation of certain sender identities did not reflect the change in the email volume received from them. A constant spam rate does not imply that the volume of email is constant. For example, consider a spammer who propagates 1 spam email out of 10 emails in the first interval (spam rate = 0.1, reputation of 0.9) followed by 900 spam messages out of 1000 emails (spam rate = 0.9; reputation = 0.1) in the second interval. In this case, with a value α as 0.5, the reputation would be 0.5 (an average of 0.1 and 0.9). However, such a sender should be penalized more.

To track sender's reputation more closely, more emphasis should be placed on the interval in which the email volume was higher. For example, if the email volume in the past interval was higher than the email volume in the present, more emphasis should be placed in the past. Likewise, when the email volume in the present is higher, the present reputation should be considered more than the past reputation.

Incorporating the change in the email volume on a global scale requires all the RepuCollectors to share both peer-reputations and the email volume. Sharing of the email volume invokes potential for further attacks on the reputation framework; for instance, some receiver organizations could provide incorrect volume information about sender identities to increase/decrease their reputations. The initial deployment showed that the

$Given\ Vol_{m-1}, Vol_m\ represents\ volume\ of\ email\ received\ in\ intervals\ m-1, m.$
$GR_{m-1}, GR_m\ \ represents\ email\ good\ rate\ received\ in\ intervals\ m-1, m.$
$Multi\text{-}Frac\ represents\ Multiplicative\ factor\ a\ constant$

$$\begin{aligned}
&If\ (Vol_{m-1} \geq Vol_m)\\
&\quad \text{Vol-Enh GR} = \left(\left(\frac{\mathrm{Vol_{m-1}}}{\mathrm{Vol_m}}\right) \times \mathrm{GR_{m-1}}\ + GR_m\right)\\
&else\\
&\quad \text{Vol-Enh GR} = \left(\left(\frac{\mathrm{Vol_m}}{\mathrm{Vol_{m-1}}}\right) \times \mathrm{GR_{m-1}}\ + GR_{m-1}\right)\\
&endif \qquad\qquad\qquad\qquad \dots\dots\dots\dots\dots \text{[Equation 4.5a]}
\end{aligned}$$

$$\alpha' = 1 - \ e^{-(Multi\text{-}Frac \times \text{Vol-Enh GR})} \qquad \dots\dots\dots\dots\dots \text{[Equation 4.5b]}$$

$Where$ Vol-Enh GR is the Volume-Enhanced GoodRate
$\alpha' is\ the\ instantaneous\ value\ of\ \alpha$

Equation 4.5: Instantaneous value of α based on the volume of email received at a RepuServer.

majority of sender identities were spammers. As incorporating email volume at a global level, participating receiver organizations could lie about the volume sent by a sender. For this reason, email volume should be incorporated at RepuServers but not at the RepuCollectors.

Any incorrect volume embedded into reputations at RepuServer would only be constrained to the organization. Such incorrect reputation-view from a receiver organization will not significantly affect the global reputation since such data will be refuted by other honest receiver organizations.

To incorporate email volume as a basis for the computation, an exponentiation is selected due to its monotonic property. Due to this property, the e-x always lies in the interval (0, 1) and is a monotonically decreasing function. A monotonically decreasing function is required as the value of α should decrease as the volume rate increases.

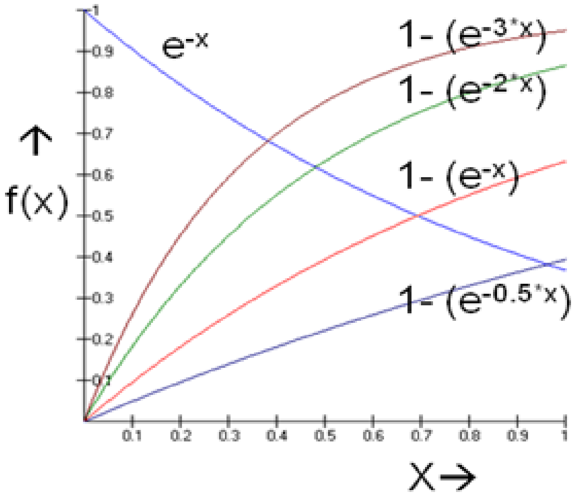


Figure 4.2: The function e^{-x} function is monotonically decreasing function. As the value of Volume-Enhanced Good Rate increases, a higher curve is selected resulting in a larger value of α.

Equation 4.5 demonstrates the mechanism to compute the instantaneous correlation factor α based on the email volume. Equation 4.5a shows the mechanism to compute Volume-Enhanced Good Rate (Vol-Enh GR) is the sum of the good rate in the interval that had larger volume and a fraction of the good rate in the other. This implies that having the Good Rate (GR) constant, if the volume in present is large, the Vol-Enh GR is the sum of good rate in the present and a fraction of the good rate in the past. If the volume in the past is small, the good rate in the past is small multiplied by the factor past volume divided by present volume. Equation 4.5b is used to compute instantaneous α, a lower value of Vol-Enh GR relative to the present good rate leads to a higher instantaneous α to accurately compute reputation.

Figure 4.2 demonstrates that larger the value of Vol-Enh GR implies a different curve is selected resulting in a larger value of α. High α implies a greater importance is

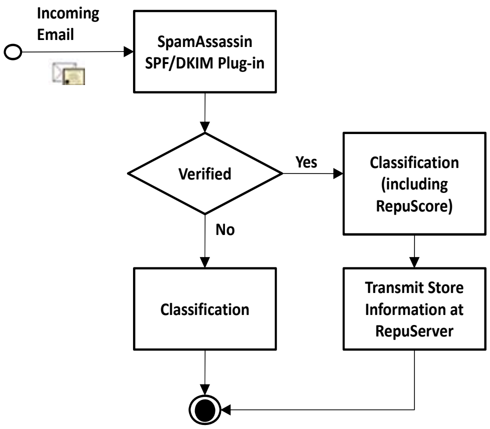

Incoming Email
SpamAssassin SPF/DKIM Plug-in
Verified
Yes
No
Classification (including RepuScore)
Classification
Transmit Store Information at RepuServer

Figure 4.3: SpamAssassin plug-in collects statistics from mail servers and transmits it to a RepuServer. The plug-in uses other SpamAssassin plug-ins to identify sender identities.

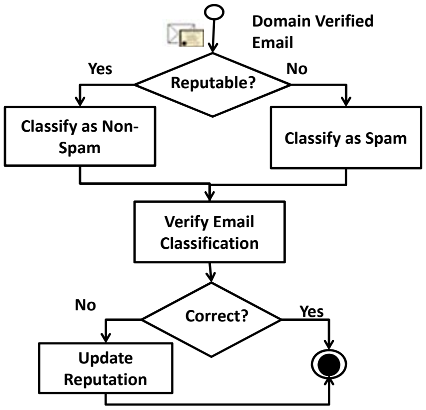


Figure 4.4: Email classification using the SpamAssassin plug-in. Email verification can use other spam classification techniques to correct the reputation-based classification.

placed on the past interval as compared to the present interval. Likewise, high Vol-Enh GR leads to a lower value for instantaneous α. The multiplicative factor is used to decrease large values of Vol-Enh GR.

For the example discussed in the first paragraph, the Vol-Enh GR = 0.901 (1 × 0.9 + 10/1000 × 0.1). Using the multiplicative factor of 1, the Volume-Enhanced reputation will be 0.42 instead of 0.5. Using Volume-Enhanced RepuScore, the computed reputation was based on the volume of the email received. In the above example, when the email was higher in the second interval, the reputation increased slowly as compared to reputation in the first interval.

4.5 RepuScore Deployment

RepuScore has been deployed in a few organizations with the help of a deployment model. The gathered votes, from receiver organizations, are collected based on user inputs or email classification programs such as SpamAssassin.

4.5.1 SpamAssassin Plug-in

A SpamAssassin plug-in that collects information about each authenticated email; i.e., whether or not an email is spam, and computes reputation for the sender identity. The RepuScore plug-in uses the available standard SpamAssassin plug-ins for SPF and DKIM to identify the senders.

Figure 4.3 demonstrates the design of the SpamAssassin plug-in that collects information for each email from an organization's mail server. Figure 4.4 shows the mechanism used to classify email after sender verification. A reputable sender's email is classified as non-spam and vice versa. Reputation-based email classification requires a feedback mechanism for checking the accuracy of classification with the help of low-process intensive mail filters. As the RepuScore plug-in has already performed the sender identity checks, content-based filters can be utilized. The information is then transmitted as a UDP packet and stored at a local RepuServer.

System administrators can select any low-process intensive email classification technique to correct the information. Such a mechanism allows high-process intensive mechanism to be used to classify emails without an associated sender identity. This allows a faster email classification when a huge volume of email is received.

The RepuServer's server module (a Perl module) maintains multiple forked instances to keep a few "hot" instances in memory to handle the normal load, while having the ability to fork a few additional instances based on the need. These processes capture the packets transmitted to them by the RepuServer client module and write the incoming data into a MySQL database. A cronjob initializes a script that computes the reputation at every reputation interval by invoking SQL statements.

CHAPTER 5: RESULTS

Our results for the projects P-Messaging (Kang 2006) and Central RepuScore algorithms (Singaraju 2007, Singaraju 2008) are published at USENIX LISA 06, LISA 07 and CEAS 08 conferences. In addition, this dissertation also discusses the results for IP based reputation and False Positives and False Negatives.

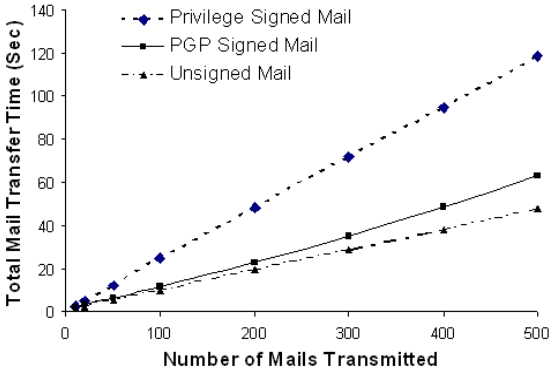
5.1 P-Messaging Evaluation

P-Messaging technology can be developed in Java and can be provided with a command-line interface for configuring the server. P-Messaging can be evaluated based on the ease of configuring the servers. Other experiments demonstrate the performance costs associated compared with PGP-signed email:

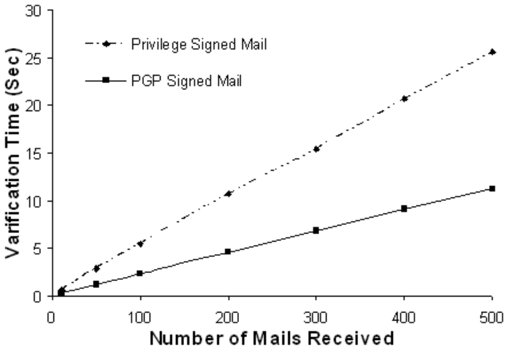
1. Signature Generation and Verification Time
2. Privilege Generation and Verification Time

5.1.1 Signature Generation and Verification Time

To demonstrate the overhead incurred due to generation of P-Tags, P-Messaging's tag generation performance is compared with the time taken to generate PGP digital signature and unsigned emails. Graph 5.1 shows the results. The time taken to generate the tag was reasonably higher than for PGP and unsigned messages. This overhead includes the time taken to request a privilege including two signature generations. As P-Messaging uses a double signature as compared to one signature for PGP, the results shown are expected.



Graph 5.1: The overhead of P-tag using P-Messaging compared to PGP signed and unsigned emails.



Graph 5.2: The overhead of P-Tag verification compared to PGP signature verification.

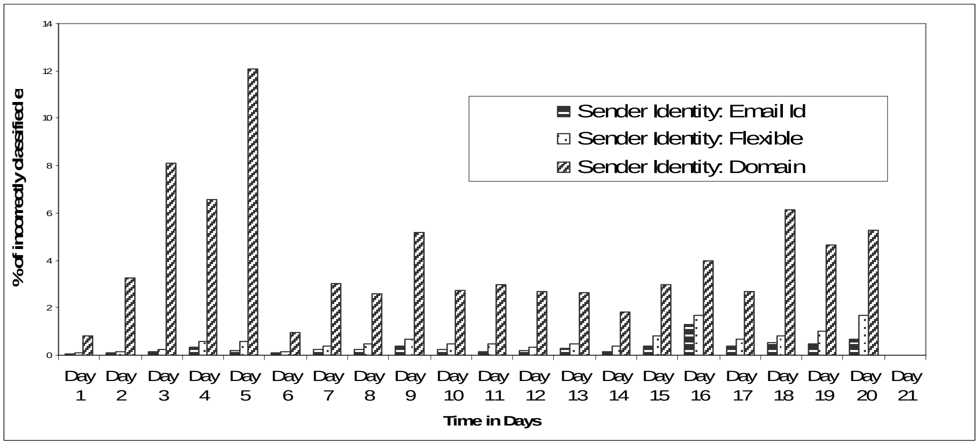
The overhead incurred due to verification of the privilege is compared with the time taken for verifying a PGP signed message. The results are shown in Graph 5.2 where the time taken to verify the emails is twice the time taken to verify the PGP signed mail that is expected again.

5.1.2 Privilege Generation and Verification Time

The experiments show that the time taken to generate a P-Tag was about 0.16 seconds. This time included the time taken to generate double signatures: one for the privilege and another for the P-Server server. The time taken to verify a message was about 0.09 seconds, again this time involved the time taken to verify the P-Server signature, and then the Privilege signature. It also involved the time to retrieve the privileges' public key from the sender's P-Server.

5.1.3 Effect of Cardinality on Collaborative Blacklists

This section discusses the effects of cardinality of sender's identity and the number of collaborating receivers on the accuracy of blacklists. This section begins by discussing the Log Statistics obtained from a global non-profit organization.

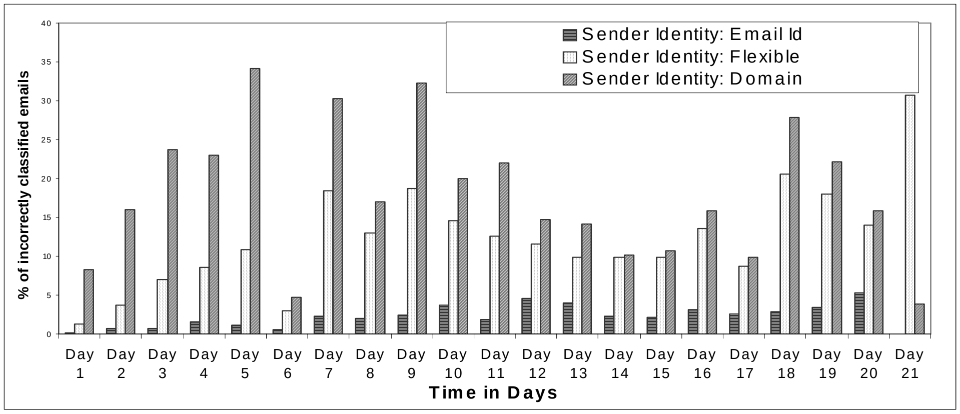


Graph 5.3: Percentage of incorrectly classified email per day without receiver collaboration to maintain a blacklist.

Log Statistics

The experiments were performed using 21-day logs [from Aug 6th - 26th 2006] obtained from a global non-profit organization that maintained 5 domains. Each of the domains additionally maintained multiple sub-domains. The organization ran a Postfix mail server (Hildebrandt 2003). The mail trace contained 41,400 domains that sent emails to the organization and about 1,133 domains to which emails were sent. The email trace contained 271,200 emails. Of these, 150,951 emails, i.e., 55% were marked as spam with the help of Real Time Blacklists (RBL) (Real Time Blackhole 2002).

The spam emails identified by RBLs were marked as ‘rejects’ with the reason for reject and the RBL that has identified the sender as a spammer. The mail server also rejected mails if the sender domain did not exist. The user behavior is modeled based on the results from the RBL. In the event that a reject was noted, the RBL rejects were used to simulate user input to blacklists.



Graph 5.4: Percentage of incorrectly classified email per day where receivers within an organization collaborated to maintain a blacklist.

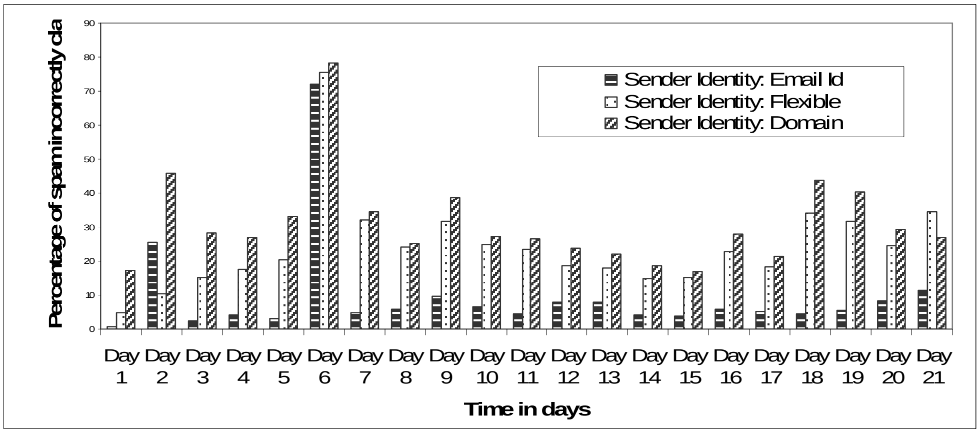
Rejects from RBL Lookup Simulates User Behavior for Blacklists

The first experiment demonstrates that the email rejects noted in the mail log due to RBL lookup can be used to simulate user behavior. To demonstrate user blacklist behavior, every user is assumed to maintain a blacklist for every sender's email id as an identity. Upon receipt of an email, the receiver's local blacklist is checked. If the sender has not been blacklisted, the RBL entry is checked to simulate the user's entry. If RBL marks the email as spam, the sender is added to the blacklists.

The experiment showed that about 510 emails (or 0.18%) were false positives (legitimate emails incorrectly recognized as spam) and false negatives (spam incorrectly recognized as legitimate email). A deviation of about 0.18% indicates a strong correlation between RBLs and local blacklist mechanism.

Effect of Cardinality of Identity and Receiver Collaboration

The mail logs described in Section 5.2.1 was used to demonstrate the effectiveness of blacklist mechanism. The receivers maintain blacklists of senders with the help of a shared blacklist among the receivers, based on the senders' identities. The



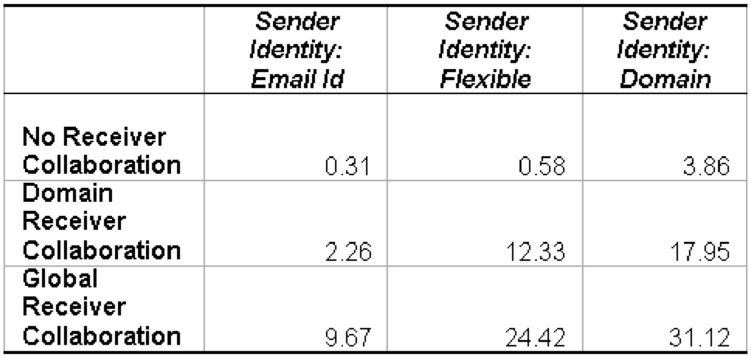
Graph 5.5: Percentage of incorrectly classified email per day where all 5 organizations collaborate to maintain a global Blacklist.

experiment is similar to the one performed in Section 5.2.1. When an email is received, the identity is checked against the local blacklist. If the identity is present in the blacklist, the email is rejected. If not, the user reads the email and the user determines whether it is spam. When the user determines that the email is spam, the user puts the sender identity on the local blacklist. The rejects logs from RBL are employed to emulate user behavior.

These experiments use three types of sender identity: email id, domain name, and flexible identity. The cardinality of the sender identity is selected as the number of users associated with the identity. (e.g., the cardinality of sender identity using single email id is 1.) In the experiments, flexible identity contained at least 2 randomly selected email ids, thus its cardinality is larger than that of single email id and smaller than that of the domain as sender identity.

Assuming that the RBL simulates user behavior, Graph 5.3, Graph 5.4 and Graph 5.5 denote the number of incorrectly classified emails per day plotted against the number of days. Incorrectly-classified emails include both false positives and false

Table 5.1: The average percentage of incorrectly classified spam over 21 days as a function of receiver collaboration and sender identity.



	Sender Identity: Email Id	*Sender Identity: Flexible*	*Sender Identity: Domain*
No Receiver Collaboration	0.31	0.58	3.86
Domain Receiver Collaboration	2.26	12.33	17.95
Global Receiver Collaboration	9.67	24.42	31.12

negatives. Graph 5.3 shows the amount of emails that were incorrectly recognized for different sender identities without any receiver collaboration. It can be seen that as the cardinality of the identity increases, the amount of incorrectly classified emails increased.

Graph 5.4 demonstrates the amount of incorrectly recognized spam when the users among a domain collaborate. The graph reiterates the same behavior as in Graph 5.3, where as the cardinality of the identity increases, the amount of incorrectly classified emails increase. Comparing this graph with Graph 5.5, it can be seen that by introducing receiver collaboration, the percentage of incorrectly recognized emails has increased.

Graph 5.5 demonstrates the number of incorrectly classified emails when the senders maintained a global blacklist. The global blacklist was maintained among the 5 domains. Table 5.1 summarizes the results, showing the overall percentage of incorrectly classified emails as a function of the cardinality of the sender identity and receiver collaboration. For highest accuracy, the receivers should not collaborate when the sender identity being an email id. The most inaccurate method for maintaining a blacklist is with sender identity being a domain with global collaboration.

Even though accuracy is highest for email id as identity, it entails a huge amount of storage and collaboration requirements. On the other hand, collaboration for the sender identity being set as domain is highly inaccurate. Additionally, without receiver collaboration, the time required to classify emails for the users is huge. This suggests that collaborative systems should use some sort of collaboration for a flexible identity. The flexible identity should be fine-granular than the domain and coarse-granular than the email id. Such an identity would be ideal to reduce the amount of incorrectly recognized email based on sender identity.

5.2 RepuScore Evaluation

The effectiveness of RepuScore is demonstrated through experiments with the help of:

a) Simulated logs to demonstrate specific properties of RepuScore; and

b) Deployed results from RepuScore

Simulated logs

The simulated logs were created with 100 RepuCollectors spanning 45 intervals. A random number of RepuServers is selected which reported to their local RepuCollectors. The number of emails and spam that were transmitted to and from an organization was perturbed using a random number; for example, since RepuScore creates a trusted group of reputable senders, the spam rate among them was set at under 20%, whereas a spamming domain’s spam rate was set at greater than 95%. (This trend is seen in the logs from the non-profit organization.)

Mail logs from a Non-Profit Organization

The logs were 20-day mail logs collected by 5 domains maintained by the

Table 5.2: Sample mail logs from Postfix Server

```
Aug 20 04:32:22 abc postfix/smtpd[17111]: < unknown[someIP]: RCPT
TO:<someone@somewhere.com>
Aug 20 04:32:22 abc postfix/smtpd[17111]: extract_addr: input: <someone@somewhere.com>
…
Aug 20 04:32:22 abc postfix/smtpd[17111]: >>> START Helo command RESTRICTIONS <<<
Aug 20 04:32:22 abc postfix/smtpd[17111]: generic_checks: name=permit_mynetworks
Aug 20 04:32:22 abc postfix/smtpd[17111]: permit_mynetworks: unknown someIP
Aug 20 04:32:22 abc postfix/smtpd[17111]: match_hostaddr: someIP ~? 127.0.0.0/8
Aug 20 04:32:22 abc postfix/smtpd[17111]: match_list_match: someIP: no match
Aug 20 04:32:22 abc postfix/smtpd[17111]: generic_checks:
name=reject_unknown_sender_domain
```

Table 5.3: Sample logs from simulated RepuServer table

Interval	Sender	Spam	Total
1	43	319	1227
1	96	1575	6058
1	32	549	1962
1	32	481	2294
1	50	626	3915

Table 5.4: Sample RepuCollector and CA reputation tables

Interval	Sender	Reputation
1	1	0.827423
1	2	0.871215
1	3	0.840456
1	4	0.832936
1	5	0.829092

organization. It contained about 45,000 domains to which 450,000 emails were sent, 55% were marked as spam by RBLs or rejected since the sender domain did not exist through reverse DNS lookup; a reverse DNS lookup checks if the sender is likely using an inexpensive internet service, such as a dialup. A sizeable amount of spam originating from inexpensive dial-up Internet services is rejected due to reverse DNS lookups.

Deployment at Three Organizations

During the deployment for 174+ days computed reputations for 16,509 sender identities authenticated using SPF and DKIM. Minimum Good Reputation is defined as the minimum reputation to be considered a credible sender. For the experiments, a Minimum Good Reputation is selected to be 0.5 to classify the emails and discuss the reasons for selecting the same. Lifetime of a sender identity is defined as the number of reputation intervals between the first and the last occasion including the first occasion the sender identity sent an email. For example, if the sender appears just on one day, the Lifetime is considered 1. For the experiments, a value of α of 0.8 is selected for original RepuScore for all comparisons. The value of 0.8 is selected to allow more importance in the past than the present.

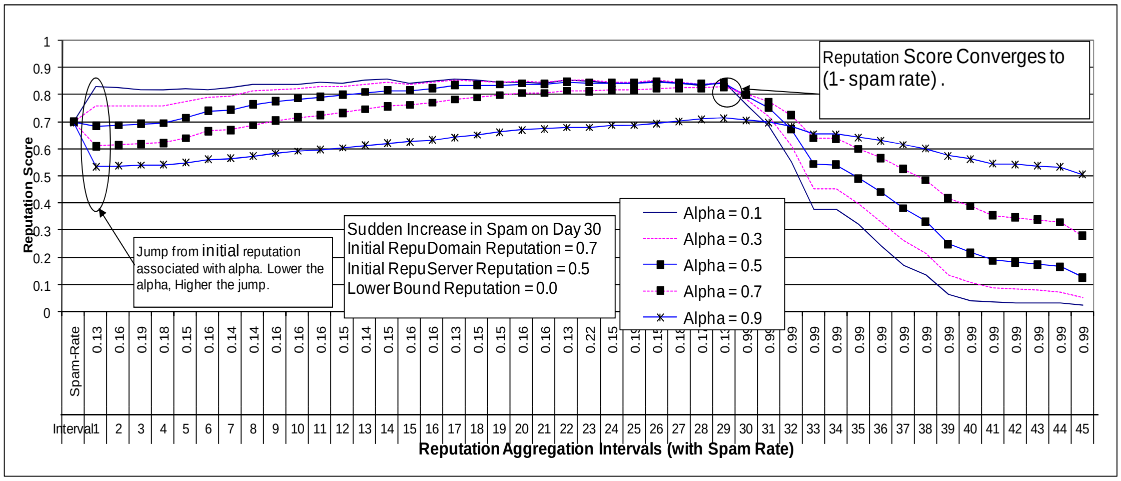
The deployment experiments show information from two organizations. The first receiver organization is a small business organization with a user base of 50 that uses SpamAssassin to classify the emails. This organization has deployed RepuScore since 10/9/2007.

The second receiver organization is an Email Service Provider (ESP) that has deployed RepuScore since 2/7/2008. The organization has 78,000 users of which about 10,000 paying customers have SpamAssassin plug-in to identify senders. About 17,000+ verified authenticated emails are received by the organization in a single day.

5.2.1 Format of Logs to RepuScore Algorithm

The mail logs from the non-profit organization were based on postfix mail server. Table 5.2 shows the different checks that the postfix server performed. The mail server uses spamhaus.org as the RBL. The anonymized logs captured received emails for 5 organizations.

The RepuServer computes the information from different senders. This is maintained in the format shown in Table 5.3. The RepuCollector and the Central Authority maintain the reputation for senders in the format shown in Table 5.4.



Graph 5.6: The change in the reputation of a trusted domain that transmits spam after reputation interval 30 as a function of α. Reputation eventually converges to (1 - average spam rate) after multiple reputation intervals. A high value of α places importance to reputation in the previous interval, whereas a lower value places higher importance to the present interval. For a high value of α, it takes a long time for the reputation to change whereas for low α value the change is faster. The sudden change from the initial score in the first interval is due to initial sender reputation set at 0.7.

5.2.2 Experiments with Simulated Logs

This section discusses the results from the simulated logs:

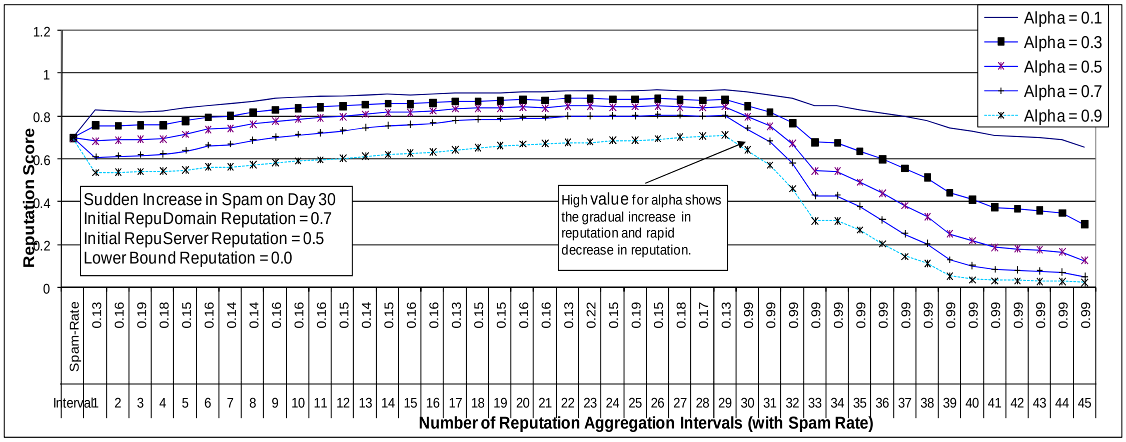
a) Effect of α on trusted Sender with sudden increase transmitted spam

b) Initial Values for RepuCollector

c) Resilience to Sybil Attacks

5.2.2.1 Effect of α on trusted Sender with sudden increase transmitted spam

This section discusses the scenario where spammers build reputation and suddenly transmitting huge amounts of spam. In such cases, it is expected that the sender identities' reputation would be high until the spamming activity, after which the reputation would decrease quickly. To demonstrate the effectiveness of RepuScore, logs with 100 RepuCollectors spanning 45 aggregation intervals were created. Random number of RepuServers are selected which reported their reputations to their local



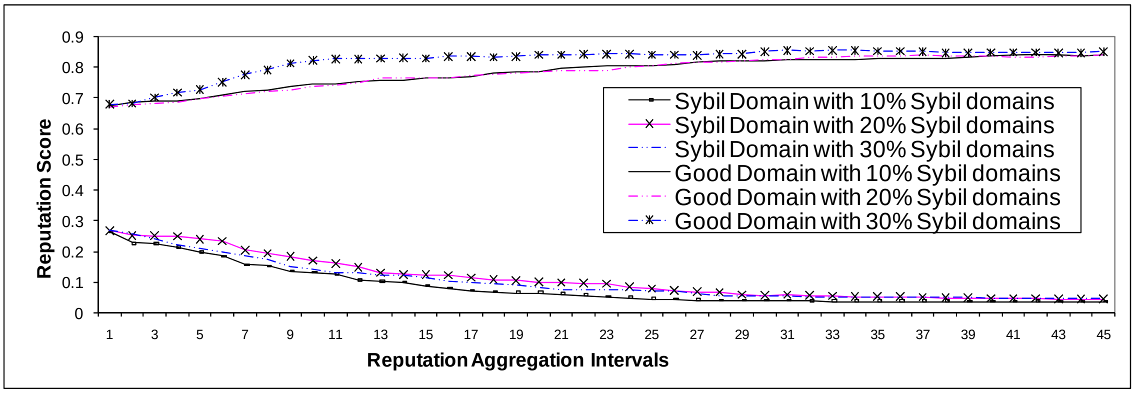
Graph 5.7: In the modified RepuScore algorithm, a high value of α (other than 1.0) implies gradual increase, but fast decrease in reputation.

RepuCollectors. The number of email and spam transmitted to and from an organization was perturbed using a random number; for example, since RepuScore for trusted and reputable senders, the spam rate among them was set at under 20%, whereas a spamming domain's spam rate was set at greater than 95%. (The trend from real world logs.)

Using Equation 1 without Swapping α and (1- α)

Graph 5.6 demonstrates the reputation of a sender identity from which the amounts of spam suddenly increased and demonstrate these using different values of α. For the first 30 aggregation intervals, the RepuCollector built its reputation and attempted to be a trusted sender. After aggregation interval 30, the spam rate from the RepuCollector increased to 95%. The initial jump in the reputation is due to the value of α combined with the initial reputation value of RepuCollector that was set at 0.5. Therefore, the reputation of the RepuCollector for α = 0.9 decreased from 0.7 after the first aggregation interval.

In the case where the sender does not propagate spam, the reputation should



Graph 5.8: RepuScore's resilience towards Sybil attack. Multiple spamming domains (under a Sybil's control) increase their reputation of each other and decrease reputation for others after the reputation aggregation interval 30. Sybil domains reputations stays low and non-Sybil's reputation remains high.

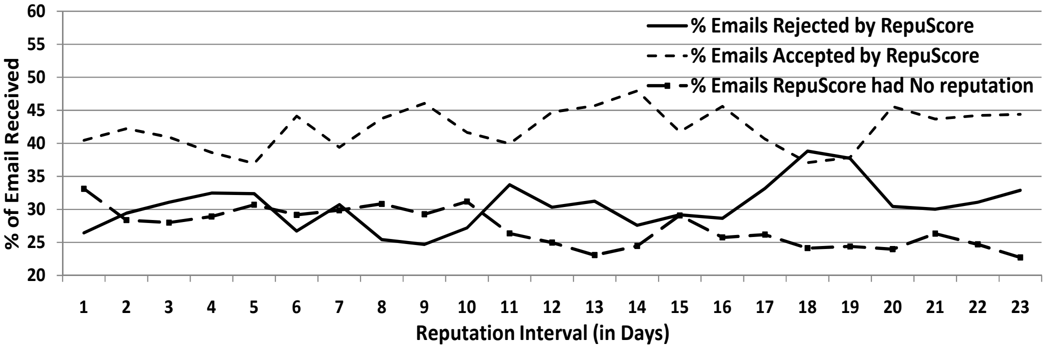
increase slowly indicating a long history of non-spamming. Hence the high value of α is good to maintain an association for a long history of good actions. If the sender propagates spam, the reputation should decrease immediately, reflecting the current actions of the sender. A low value of α guarantees an immediate reduction when the sender propagates spam.

Using Equation 1 with Swapping α and (1- α)

With the interchange of α and (1- α) in Equation 1, allows the slow increase and fast decrease in the reputation. Graph 5.7 demonstrates the values of reputation by using the modified algorithm. For high values of α, reputation increases gradually but decreases more rapidly.

5.2.2.2 Initial Values for RepuCollector

Setting an appropriate initial value for RepuCollectors' reputation is extremely important to maintain a trusted group of reputable senders. For instance, if the initial reputation scores for the RepuCollector and RepuServers are set too high, it would take a long time for the reputation to decrease. On the other hand, if the initial reputation is



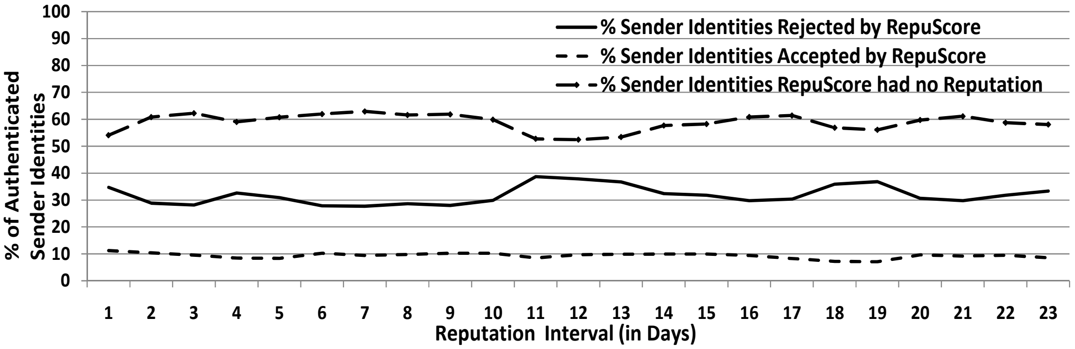
Graph 5.9: Percentage of Authenticated Emails classified using RepuScore. The reputation computed from receiver-organization 1 was used to classify emails of receiver-organization 2. On average, RepuScore classified about 72% of emails - 40% were accepted and 32% were rejected.

set too low, it would take a long time for the reputation of a non-spamming RepuCollector to increase.

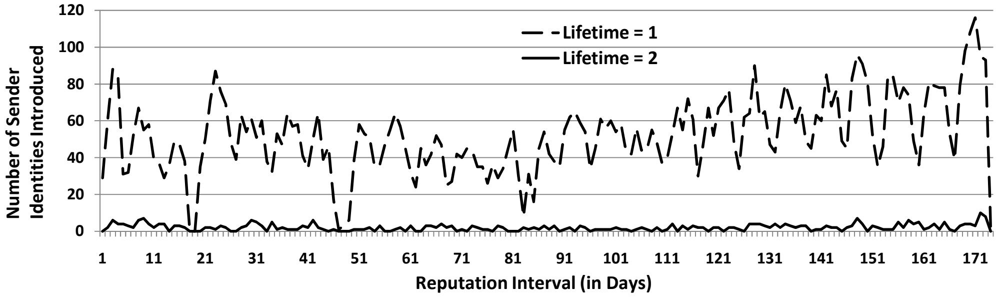
The experiments show that an ideal initial reputation value for the RepuServer and the RepuCollector is between 0.5 and 0.7. With different initial values it is noted that the average reputation of all the domains using the logs from the non-profit organization converged to about 0.6 for α = 0.1, 0.47 for α=0.5 and 0.36 for α = 0.9. Hence, an ideal initial reputation should be equal to the average reputation of all domains in the system after a long period. In order for the new reputation domains to participate in the reputation aggregation intervals, the threshold should be 0.1- 0.3 below the initial reputation.

5.2.2.3 Resilience to Sybil Attacks

To demonstrate resilience to Sybil attacks, the percentage of malicious reporting RepuCollectors is increased from 10 to 30%. Each sender transmits a high amount of spam (> 95%) for the first 30 aggregation intervals. After 30 aggregation intervals, the Sybil attackers start increasing the reputation of its own Sybil domains and decrease the



Graph 5.10: Around 10% of the authenticated sender identities were credible senders; while about 32% were known spammers. RepuScore had no reputation information for 58% of the senders.

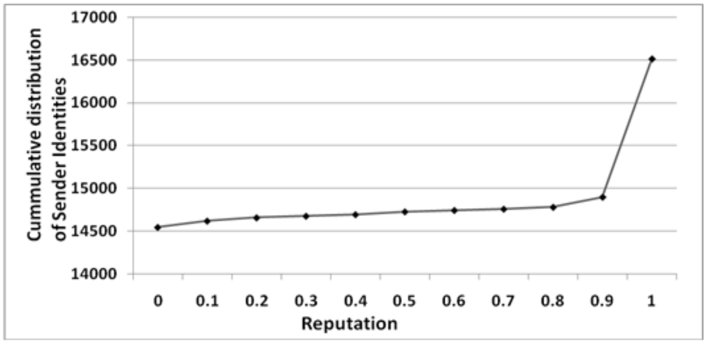


Graph 5.11: Number of sender identities with lifetime of 1 day (sent emails only on 1 day in 174 days) and 2 days (sent emails on 2 consecutive days in 174 days) plotted against their first appearance. For the sake of brevity, the number of domains which had lifetimes of 3 and above is not shown. However, it is noted that the value is negligible compared to Lifetime 1 and 2. About 8000 sender identities with Lifetime 1 sent emails were noticed.

reputation of other domains. Graph 5.8 demonstrates the results where the reputation of the Sybil domains steadily decreased, but the reputation of the non-Sybil domains increased.

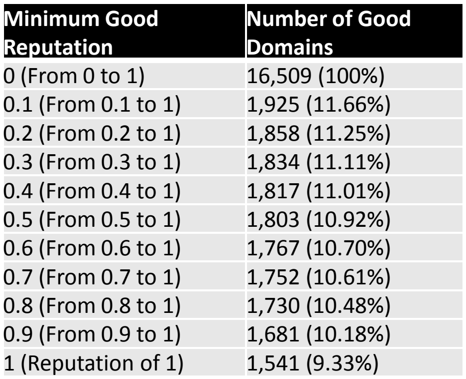
5.2.3 Results from the Deployment

The RepuScore statistics, effectiveness of RepuScore and the results of Volume-Enhanced RepuScore is demonstrated in this section.



Graph 5.12: Cumulative distribution of sender identities as a function of reputation. 97.8% of the identities had reputation of 0 or 1.

Table 5.5: The distribution of reputation as a function of minimum good reputation.

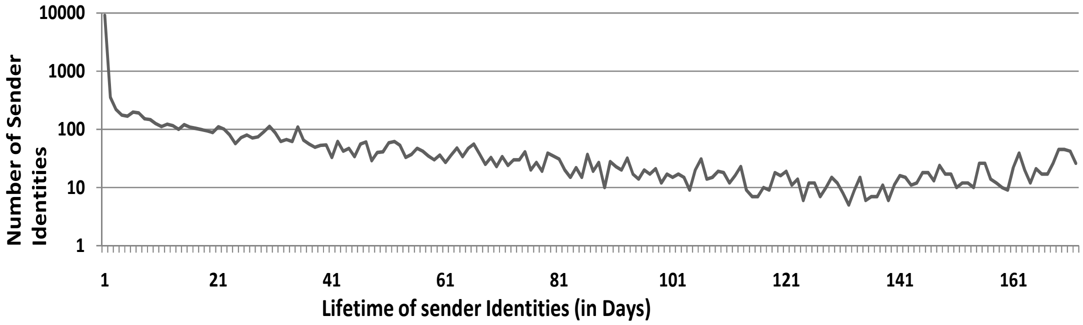


Minimum Good Reputation	Number of Good Domains
0 (From 0 to 1)	16,509 (100%)
0.1 (From 0.1 to 1)	1,925 (11.66%)
0.2 (From 0.2 to 1)	1,858 (11.25%)
0.3 (From 0.3 to 1)	1,834 (11.11%)
0.4 (From 0.4 to 1)	1,817 (11.01%)
0.5 (From 0.5 to 1)	1,803 (10.92%)
0.6 (From 0.6 to 1)	1,767 (10.70%)
0.7 (From 0.7 to 1)	1,752 (10.61%)
0.8 (From 0.8 to 1)	1,730 (10.48%)
0.9 (From 0.9 to 1)	1,681 (10.18%)
1 (Reputation of 1)	1,541 (9.33%)

Effectiveness of RepuScore

To show the effectiveness of RepuScore, the reputation computed from the first receiver organization from day 107 is used to compute the reputation at the second organization's mail logs. In these graphs, the second organization uses SpamAssassin and not RepuScore to classify emails.

Graph 5.9 and Graph 5.10 shows the effectiveness of RepuScore in classifying authenticated emails. The results show that using RepuScore, while only 10% of the sender identities were good over 23 days they transmitted about 40% of the



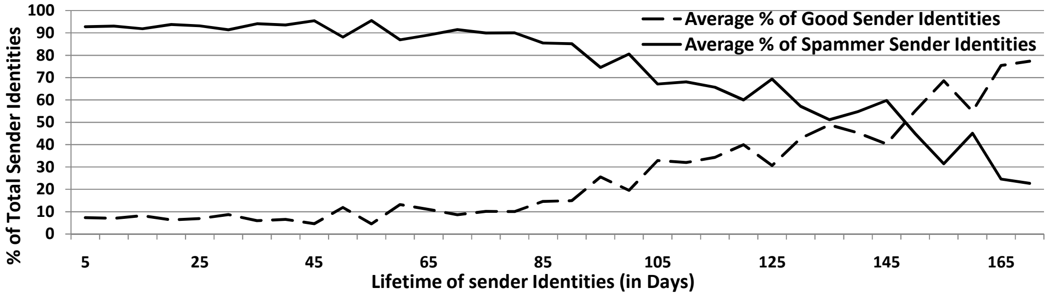
Graph 5.13: The distribution of sender identities vs. their lifetime. The number of sender identities decreases as the lifetime increases.

authenticated emails that were accepted by RepuScore. About 32% of the sender identities were spammers that sent about 32% of the authenticated emails which were rejected using RepuScore. The experiment demonstrates that with the knowledge of 42% of the sender identities, RepuScore classified 72% of the authenticated emails.

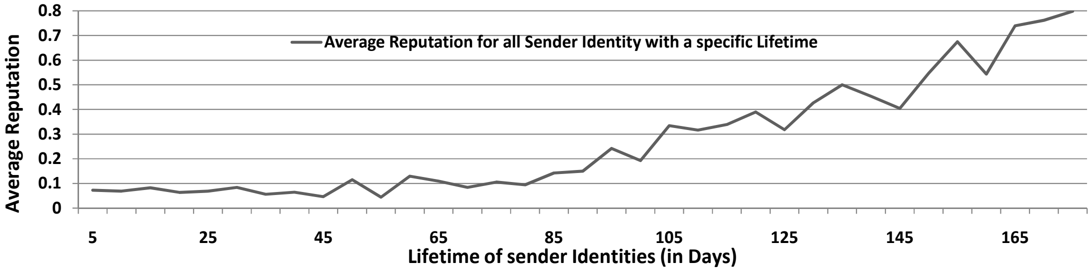
The results show that reputation gathered from a small set of users can be effective to classify emails for a large number of users. It is noticed that the number of identities RepuScore had no knowledge about was always constant indicating that a lot of new one-time sender identities were being introduced.

Graph 5.11 proves the hypothesis about a huge number of sender identities being created to spam and are taken down soon. It can be noticed that sender identities with a lifetime of 1 day are distributed over the time of the deployment. The total number of identities that sent emails only in 1 interval was about 8000. The rate at which identities sent email on two consecutive days was much lesser.

The cumulative distribution of sender identities as a function of reputation is demonstrated in Graph 5.12. Out of the 16,500+ identities, about 14,000 had a reputation of 0. The graph shows that about 97.8% of the senders have a reputation



Graph 5.14: Percentage of good (or bad) sender identities to total number of sender identities as plotted against lifetime. The probability that a sender identity being credible increases with long lifetime.

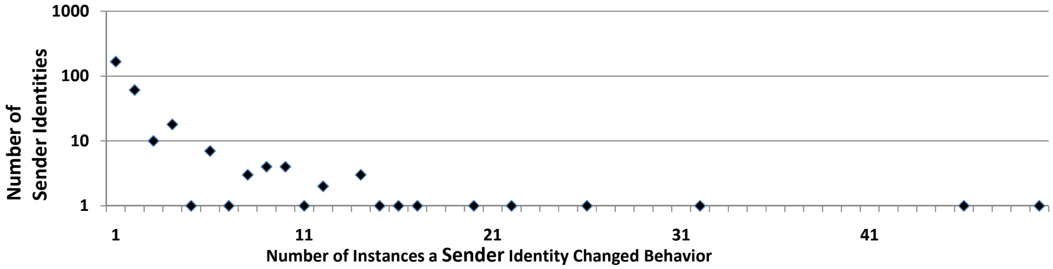


Graph 5.15: Average reputation of all sender identities with the same lifetime. As the lifetime increases, sender identity with longer lifetime has a higher reputation.

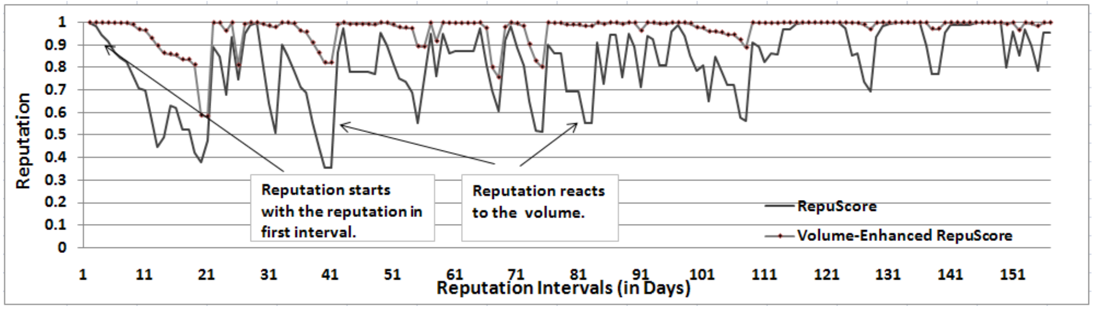
either "0" or "1". With the help of Table 5.5, a minimum good reputation to be 0.5 is selected as it bisects the two clusters with reputations of 0 and 1. With minimum good reputation as 0.5, only 10.92% of the sender identities were good senders. By changing the minimum good reputation to 0.7, 51 (0.3%) additional sender identities were considered bad.

Graph 5.13 also validates this by showing the distribution of the number of sender identities vs. their lifetime. The number of sender identities with lifetime of 1 was about 8,000. However, as the lifetime increased, the number of sender identities became smaller and evenly distributed.

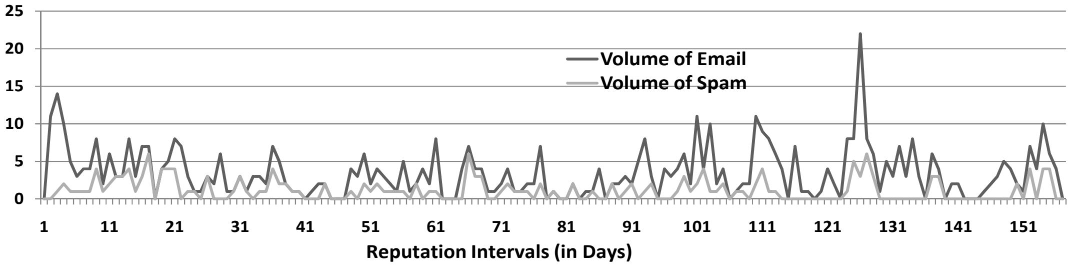
To prove the hypothesis that if the lifetime of the sender identity is long, the



Graph 5.16: Number of times a sender identity changed from good to bad or vice-versa. Only 290 sender identities (about 1.75%) changed its behavior.

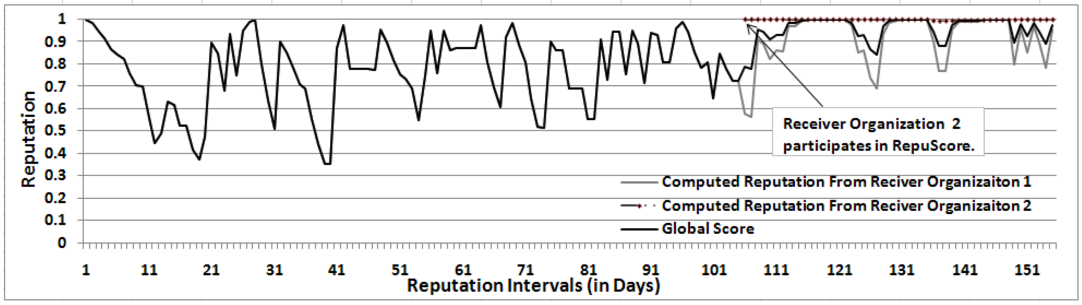


Graph 5.17: Volume-Enhanced RepuScore reacts to the email volume for a popular free email provider. After volume enhancement, the reputation between the intervals 1-11, drops radically. Reputation increases between the intervals 11-15. The slope in volume enhanced RepuScore is indicative of email volume.

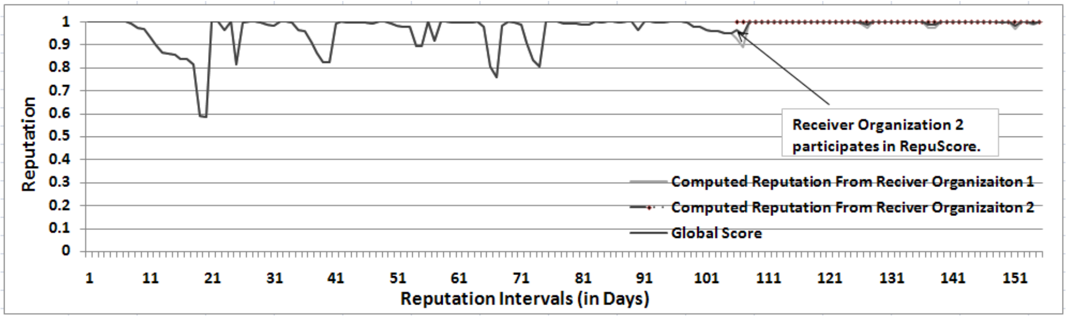


Graph 5.18: Volume of the Spam and Email noticed at receiver organization 1. The reputation of the sender identity changes based on the email volume.

probability of it being a good identity is high, a plot of the daily percentage of good and bad sender identities plotted against lifetime in Graph 5.11. The graph shows that the percentage of bad identities decreases as the lifetime increases, whereas the percentage increases for legitimate sender identity. Graph 5.15 validates this claim and shows the



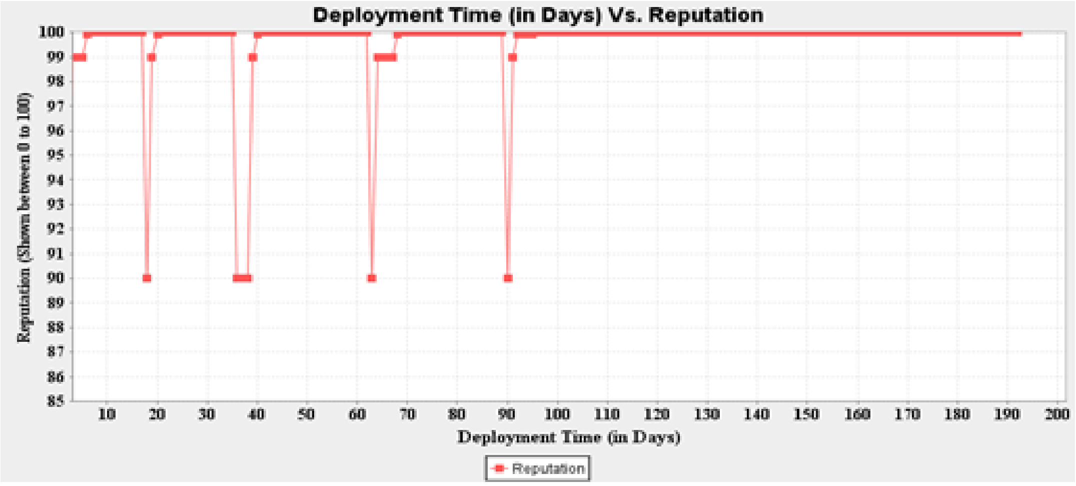
Graph 5.19: RepuScore: Reputation of the free email provider computed using information from two receiver organizations. Receiver organization 2 was introduced on day 107 of receiver organization 1.



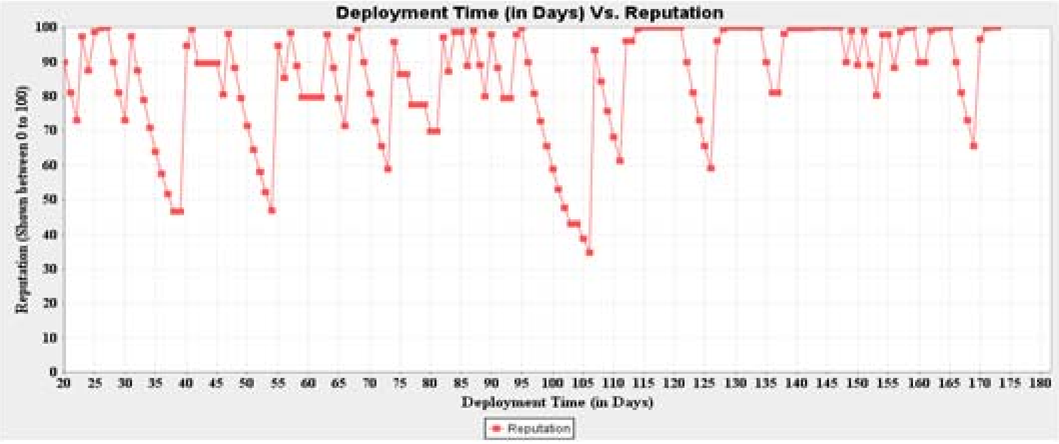
Graph 5.20: Volume-Enhanced RepuScore: Using volume, the global reputation is closer to the reputation from both the receiver organizations.

average reputations for all sender identities with same lifetimes. The curve for the average reputation for all identities shows a similar trend as percentage of good identities in Graph 5.14. Additionally, using a minimum good reputation of 0.5, credible sender identities had an average lifetime of 61.9 days while spammers had 17.47 days.

Graph 5.16 shows the number of sender identities for which the reputation changed from being good to bad or vice versa. About 1.75% changed from being good to bad or vice versa corresponding to about 291 sender identities. There were only 8 sender identities whose reputation kept changing from good to bad or vice versa more than 15 times as their reputations hovered around 0.5.



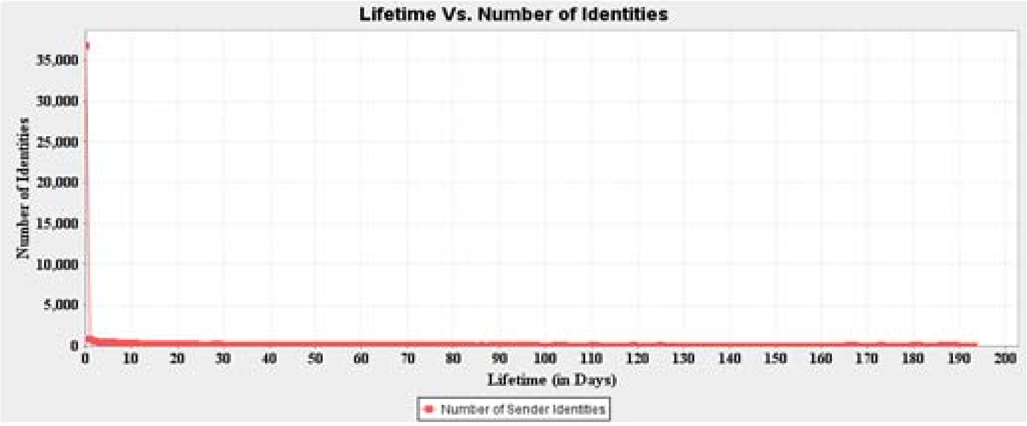
Graph 5.21: Reputation of IP address over the lifetime of the deployment.



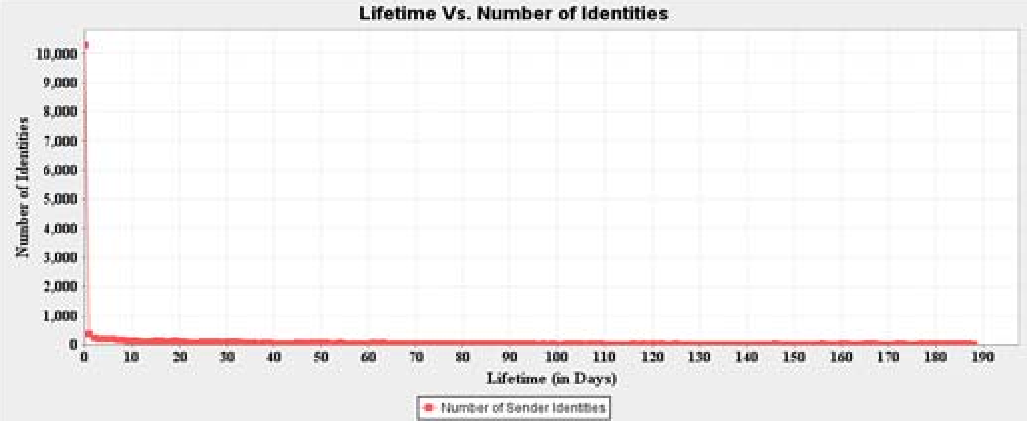
Graph 5.22: The reputation of a well known free email provider computed over 170 days.

Volume-Enhanced RepuScore

Graph 5.17 shows the reputation for a sender identity, corresponding to a popular free email service on the Internet. The sender identity had alternatively sent high and low spam rate to a single organization. Graph 5.18 shows the corresponding email volume and spam volume. Using the reputation information from the receiver organization 1, logs from the receiver organization 2 classify the emails. Graph 5.17 shows the benefit of using volume enhancement as the sender identity reputation was

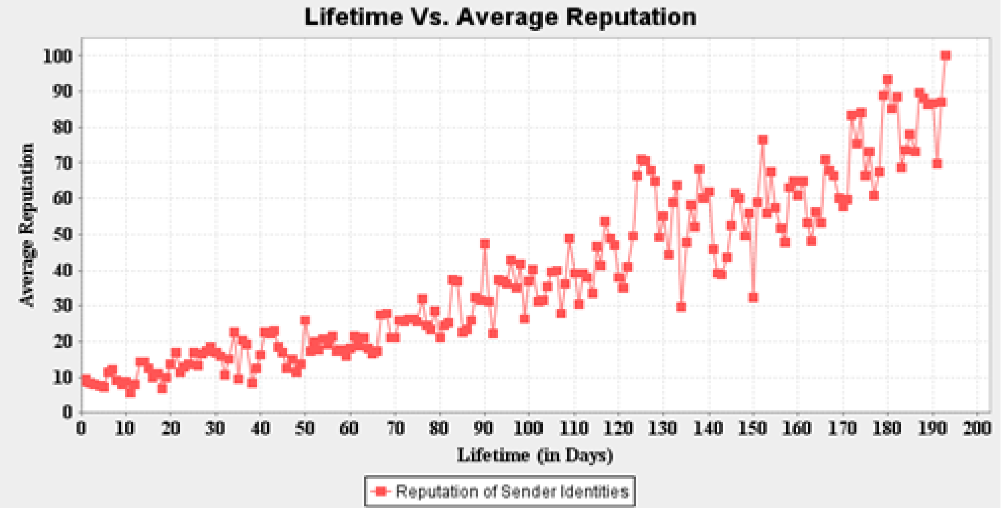


Graph 5.23: The number of IP addresses as a function of their lifetime. As lifetime increases, the number of domains decreases. A large amount of domains had a lifetime of "0".

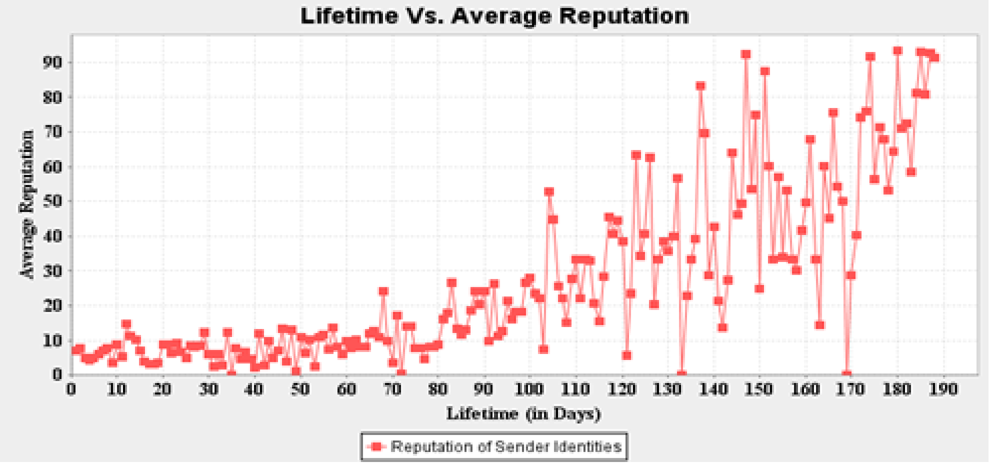


Graph 5.24: The number of authenticated sender identities as a function of lifetime. This follows the same trend as the IP addresses.

varied with the email volume. From the graphs, it is noticed that the rate at which the reputation decreases for changes based on the email volume. For example, from interval 1 to 11, the volume-enhanced reputation is lower than the original reputation. At intervals 11 to 13, the reputation computed by volume-enhancement was higher than the original RepuScore algorithm. It is noted that with the help of Volume-Enhanced RepuScore, the slope of the reputation follows the email volume. The average



Graph 5.25: The average reputation of the IP addresses as a function of the lifetime. As the lifetime increases, the average reputation for IP address also increases.

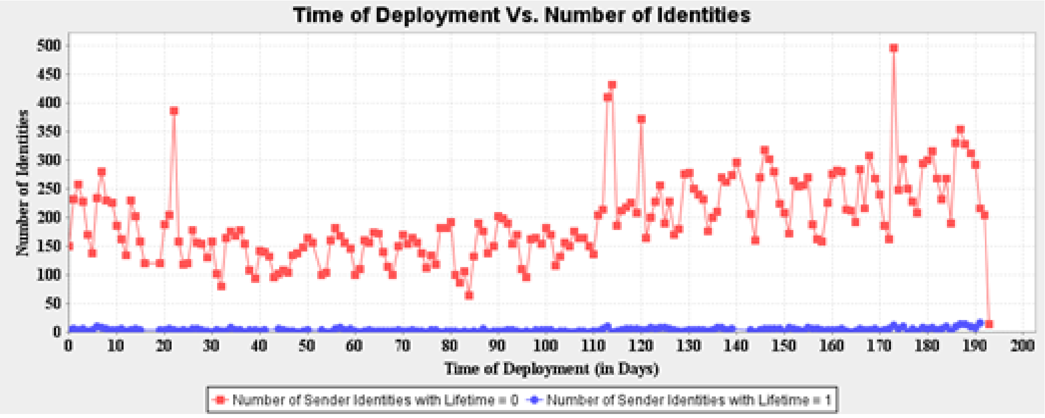


Graph 5.26: The average reputation of a sender identity as a function of lifetime. As the lifetime increases, the average reputation for authenticated sender identity also increases.

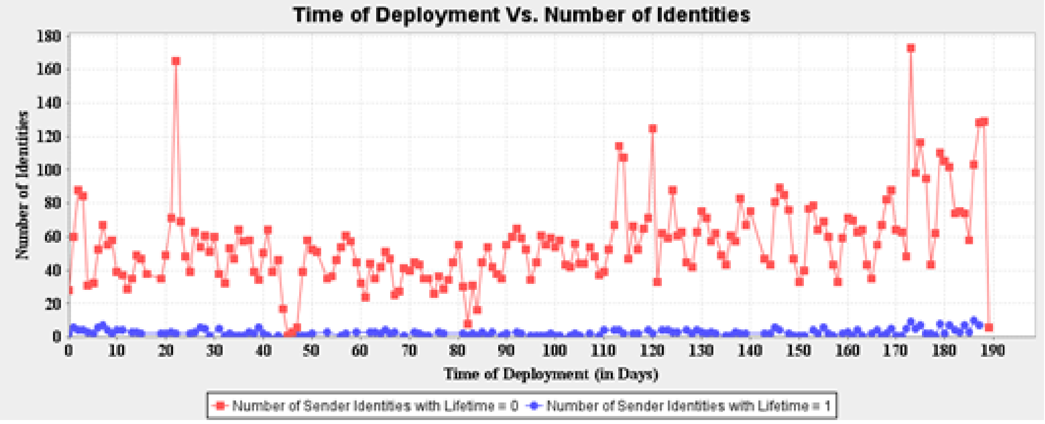
reputation over 175 days for the sender was 0.82 using the original RepuScore and was about 0.662 using volume-enhanced RepuScore. On the average over 175 days, considering a minimum good reputation of 0.5, the sender was credible.

Combining Reputations from Two Receiver Organizations

This experiment considers the effect of combining global reputation computed



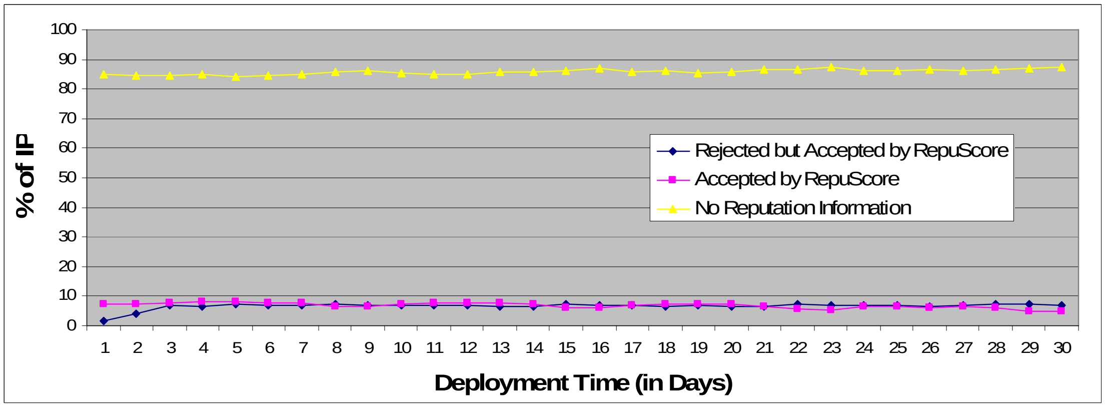
Graph 5.27: The number of IP addresses with lifetime 0 and 1 that show up during the time of deployment.



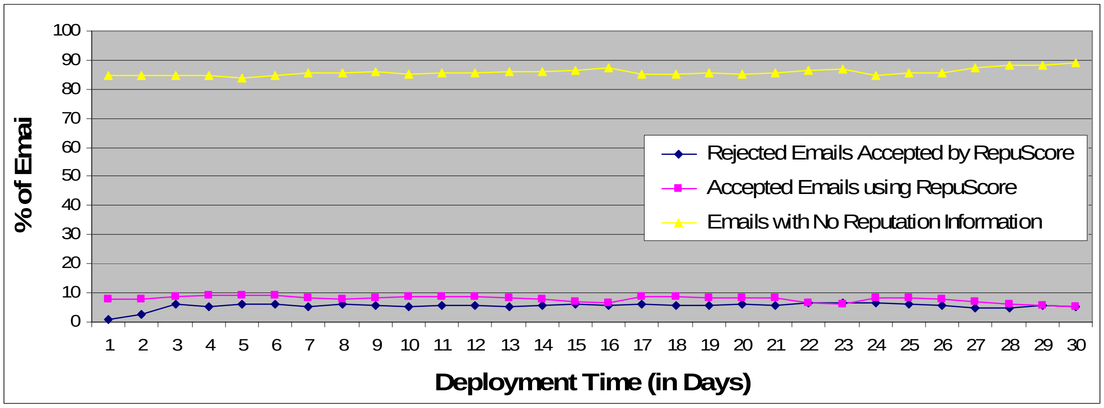
Graph 5.28: The number of sender identities that appear with Lifetime 0, 1 that appear during the time of deployment.

at two receiver organizations. For the sender identity discussed in Section 4.3, the receiver organization 2 transmitted about 38,100 authenticated emails of which 61 were spam in a span of 55 days. The receiver organization 2 started the evaluation of RepuScore from the reputation interval 107.

Graph 5.19 and Graph 5.20 shows results using RepuScore and Volume-Enhanced RepuScore. The experiments show the accuracy of the reputation depends on the number of honest receiver organizations that start contributing to RepuScore. As the



Graph 5.29: The daily percentage of IP addresses that were accepted, rejected and with no reputation.

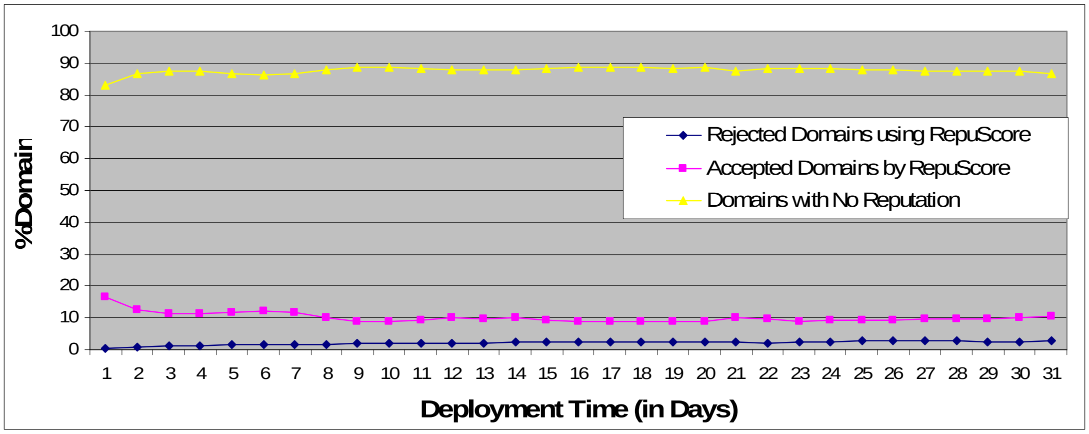


Graph 5.30: The percentage of rejected emails, accepted emails and with emails with no reputation computed by RepuScore.

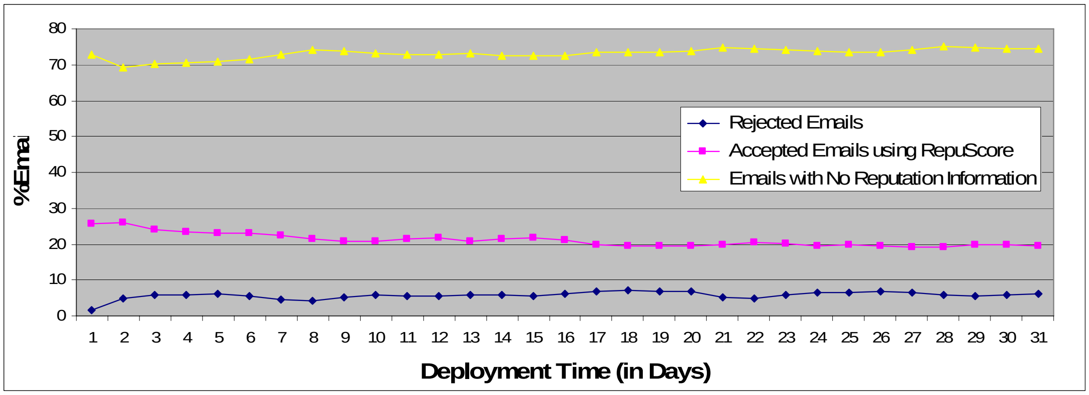
number of receiver organizations increase, the global reputation will be a weighted average of the reputation seen from different domains. In the example, as both receiver organizations did not maintain reputation for the other, the global score was a simple average.

Comparison between using IP address and Domain name as Sender Identity

Graph 5.21 shows the reputation of an IP address. The reputation is displayed against the time of deployment. It can be seen that the reputation of the sender increases



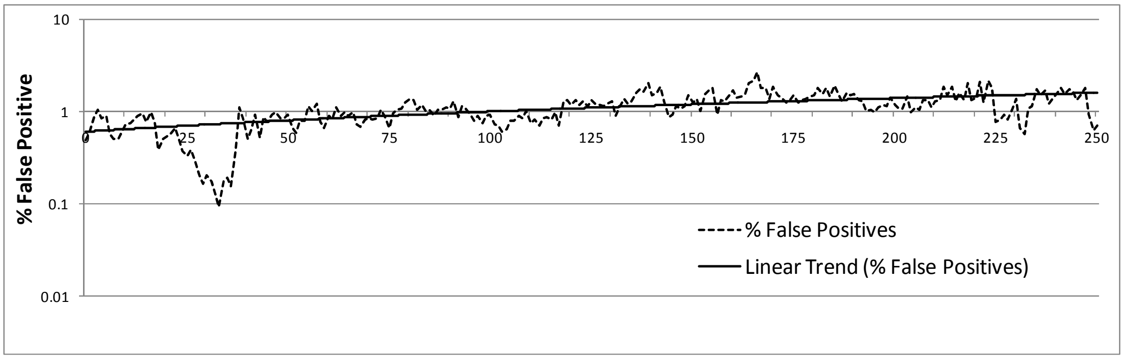
Graph 5.31: The percentage of authenticated sender identities that were rejected, accepted and with no reputation.



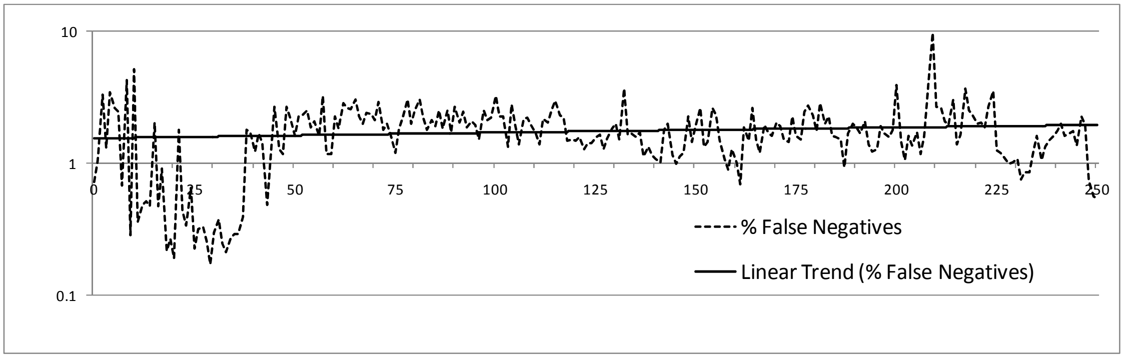
Graph 5.32: The percentage of authenticated emails classified using RepuScore.

from 90 and then becomes equal to 100. Graph 5.22 demonstrates the reputation of the sender identity of the well-known free email provider. It can be seen that as the time of deployment increased, the reputation of the sender changes a lot over the time. About 366 different IP addresses claimed to have sent email from the said domain. Of these, 184 (50.27%) had proven SPF and DKIM records.

Graph 5.23 shows the number of IP addresses having the specific lifetimes. It can be seen that as the lifetime increases, the number of IP addresses with the specific lifetime decreases. A large number of IP had a lifetime of “0”. This shows that a large



Graph 5.33: The percentage of False Positives plotted against number of days.

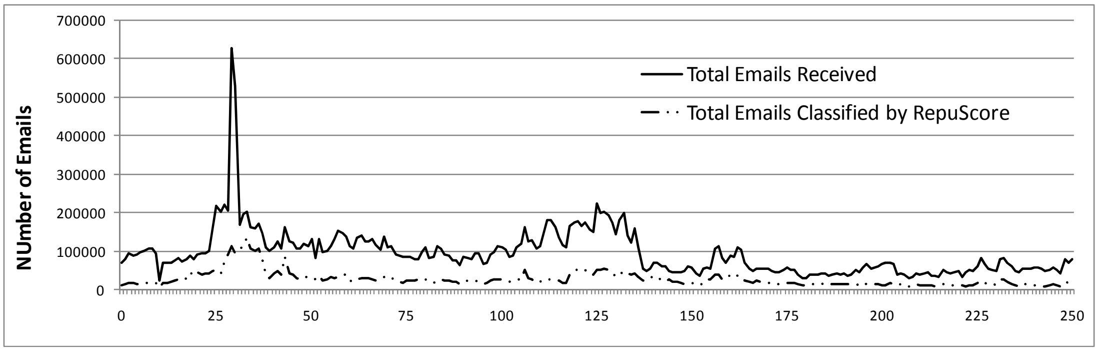


Graph 5.34: The percentage of False Negatives plotted against number of days.

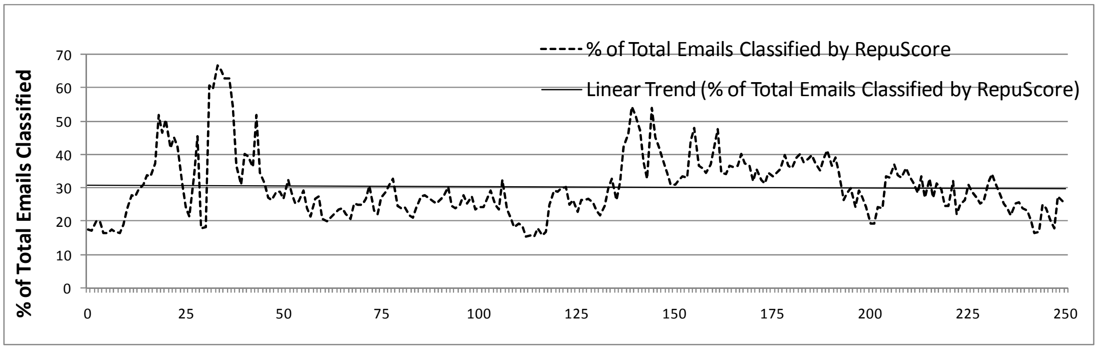
number of IP addresses come online and do not come back again. Graph 5.24 shows the same trend for domain name. Using domain name and IP address follow the same trend.

Graph 5.25 shows the average reputation of the IP addresses with the same lifetime. It can be seen that as the lifetime increases, the average reputation of the IP addresses increases. Graph 5.26 shows the same statistics for domain name. These shows that lifetime can be considered to be used for computing reputation for IP addresses.

Graph 5.27 shows the distribution of the identities over the lifetime of the deployment. It can be seen that everyday at least 200 new IP addresses transmitted



Graph 5.35: The total amount of emails received and classified using RepuScore against number of days.



Graph 5.36: The percentage of emails classified using RepuScore.

emails that did not send emails again. Graph 5.28 shows the same information for domain name.

Email Classification Using RepuScore

The reputation information is used from the single domain (16,500+ domains or 58,000+ IP addresses) and applied the information to classify emails at a second organization. The second organization received about 4.72 million emails in the month of March 2008.

Graph 5.29 shows the daily percentage of IP addresses which were classified as good, bad and those for which there was no reputation at all. RepuScore computed reputation for about 14.25% of the IP addresses that sent emails.

Graph 5.30 shows the corresponding daily percentage of emails that were classified using IP addresses for email classification. It can be seen that using IP addresses, a large amount of emails cannot be classified. This is because the bulk of emails were distributed over a large number of IP addresses for which reputation still needs to be maintained.

Graph 5.31 shows the same results using the authenticated domain names. About 31.5% of the emails were authenticated using DKIM and SPF. Graph 5.32 shows the percentage of authenticated emails that could be classified using RepuScore. With the knowledge of about 12% of the total domains that sent emails, RepuScore was able to classify about 26.8% of authenticated emails during March 2008.

Accuracy of RepuScore

Using the logs available at receiver-organization 2 with 4 mail servers, which we designate as a, b, c and d. For evaluating the accuracy of RepuScore, we computed reputation from server a, b and c over 250 days of deployment every day. Real-Time Blacklists identifies a sender as spammer or otherwise. We then used the email logs available at mail server d to calculate the accuracy at mail server d.

Graph 5.33 demonstrates the false positives due to email classification using RepuScore. We note that over 250 days of deployment, the average % false positives is about 1% of the total email identified. Graph 5.34 demonstrates the false negatives by using RepuScore. We note that the average % false negatives are about 1.2% of the total emails classified. We note that for the total emails classified, the accuracy of RepuScore is 97.8%. Graph 5.35 shows the total number of emails received by server d and the total percentage of emails classified using RepuScore. Graph 5.36 shows the %

of emails classified using RepuScore every day of the deployment. We notice that about 30.3% of the emails were classified. We believe that if server d computes reputation, the reputation can be useful in increasing the % of the emails that can be identified by RepuScore.

CHAPTER 6: CONTRIBUTIONS AND FUTURE WORK

The contributions of this dissertation work along with the future work in this direction are presented in this chapter. Section 6.1 discusses the contributions followed by Section 6.2 that discusses the future work in this direction.

6.1 Contributions

Sender accountability is a mechanism to hold senders accountable for the emails they transmit. It requires a) identifying the senders; and b) maintaining a history of the sender's activity.

This dissertation presents an authorization framework, called Privilege Messaging or P-Messaging, overlaying the existing email infrastructure while retaining the beneficial aspects such as relaying. For the sender, P-Messaging provides a mechanism that allows email delivery only if the sender possesses the privilege that the receiver would accept. Based on the privileges, the email is classified automatically according to the Privilege Tag at the receiver. Maintaining the privileges in the white-list as compared to individual email IDs allows a smaller list to be maintained. Each Privilege Server manages multiple privileges as opposed to a single credential in previously proposed domain-based authentication schemes. In case of the compromise or spam being propagated from one domain, the negative reputation is contained within a privilege rather than the complete domain.

With the help of P-Messaging, a trusted third party verifies the email's authenticity. To verify privileges, P-Messaging establishes a Circle of Trust (CoT) among the P-Server for privilege verification. P-Messaging performs dual digital signature on an email, first by the assigned privilege and then by P-Server, allowing peers in the CoT to verify the email's authenticity. This ensures that only authorized users can send messages only if their P-Server is a member of CoT, and that a P-Server needs to limit the unwanted email that it transmits or it would be revoked from the CoT.

To ensure a long history of sender accountability, this dissertation work describes, RepuScore, a collaborative reputation framework that calculates global reputation for sender identities by collecting reputation-views from multiple receivers. After being blacklisted, spammers usually adopt new sender identities. In contrast, a legitimate sender's identity typically exists for long periods. A reputation framework will be effective in blocking spam by maintaining a group of reputable trusted senders.

RepuScore distributes the overhead for reputation collection and computation by using a distributed architecture while allowing a centralized authority to collectively calculate the global reputation for each sender.

This dissertation presents algorithms to compute reputations using the history with the help of the correlation factor (α). RepuScore uses the spam rate of the senders to maintain the history of sender activity. To protect against Sybil attacks, this dissertation has devised a mechanism to consider a good sender's input more than the input submitted from a non-reputable one. During RepuScore deployment, it is noticed that the daily change in the email volume affected the reputation when only the spam-rate is used to calculate the score.

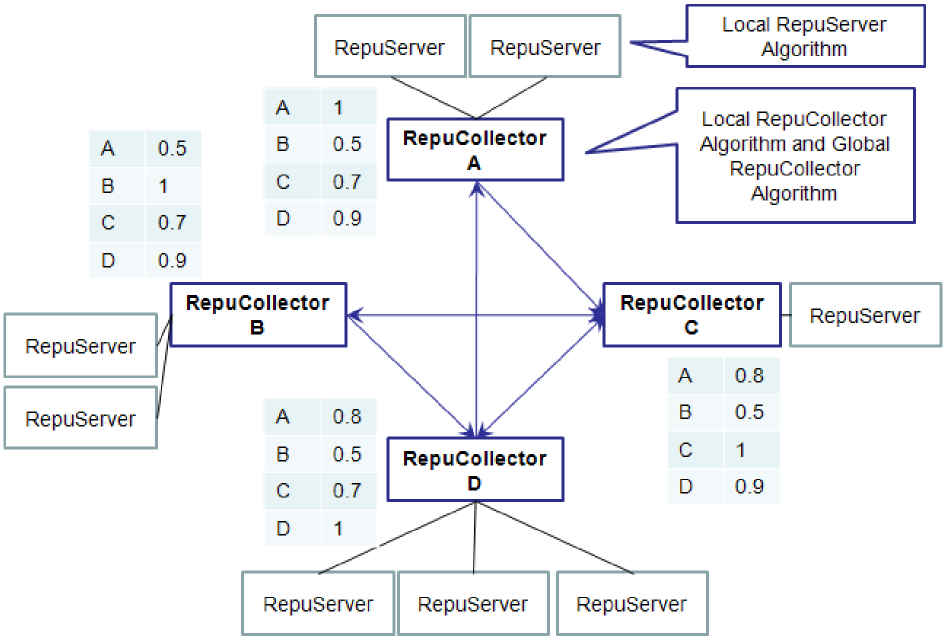
To enhance the effect of volume, Volume-Enhanced RepuScore incorporates email volume to compute reputation in addition to the spam rate of the sender identity. This dissertation also discusses a RepuScore plug-in for SpamAssassin to collect information about each email from mail servers. The plug-in is deployed RepuScore at two organizations since 10/9/2007 and computed reputations for authenticated sender identities.

The results generated from using both the IP addresses and domain name as sender identities is compared. During the deployment since October, reputations for about 16,500+ authenticated domains and about 58,000+ IP addresses have been computed. The reputation for sender identities as seen from a single domain is applied at another organization for 1 month (March 2008). The second organization received about 4.72 million emails during this period of which about 1.48 million (31.5%) were domain authenticated using SPF and DKIM. The analysis shows that RepuScore can be effective to classify sender's emails especially when the sender identity is domain authenticated.

The results show some interesting observations: a) identities with low reputation have a shorter lifetime compared to ones with high reputations; b) RepuScore was able to classify emails from about 42% of the authenticated sender identities corresponding to about 72% of the authenticated email volume; c) about 97.8% of the sender identities had reputation either near 0 or near 1. d) Average lifetime of good and bad sender identity was 61.9 and 17.47 days respectively as a large number of sender identities are created constantly that sent email only in one interval.

6.2 Distributed Reputation Management Framework

Though centralized reputation management provides high accuracy, it introduces a single point of failure. Reputation management protocol faces Denial of Service



A	0.5
B	1
C	0.7
D	0.9

A	1
B	0.5
C	0.7
D	0.9

A	0.8
B	0.5
C	1
D	0.9

A	0.8
B	0.5
C	0.7
D	1

Figure 6.1: Distributed RepuScore Architecture where RepuCollectors synchronize with friend-peers to achieve high accuracy in reputation.

attacks from spammers disabling the reputation management service. Centralized reputation management faces directed-attacks where spamming domains can maintain high reputation by sending legitimate emails to multiple domains, but send spam to a specific domain. As centralized RepuScore computes an average of the reputation, the spammer can maintain high-reputation and spam specific domains in each time-period. Organizations with low reputations can attempt to change identities when their reputation is known. This would allow organizations to use multiple domain names to send emails until their emails are accepted.

To offset these problems from Central RepuScore, a distributed version of RepuScore needs to be developed where each node has its own view of the world in comparison with a single point of view for the entire world. A single view of the world might not be amenable to all the domains in the reputation algorithm.

Given a sender identity s whose reputation is reported by Collectors with reputation $(C_1, C_2$
each having Reputations (GR_1, GR_2)*in interval* $m - 1$

```
$G_s = \frac{(C_1 \times GR_1) + (C_2 \times GR_2)}{(C_1 + C_2)}$
$if\ (s\ \in set\ of\ RepuCollectors)$
    $GR_s = G_s$
$endif$
```

Where G_s *is the global reputation of sender s in the interval m*

Equation 6.1: Distributed RepuCollector while synchronizing with peers.

Given Sync Theshold γ*: a maximum allowable difference between reputations before and a*
between at peer C_1 *with common reputation indentities* (S)*in interval m and* $m - 1$
while syncronizing with Peer C_2

```
$\theta = \left(\sum R_m - \sum R_{m-1}\right)$
$if\ (\theta < \gamma)$
  $SyncronizeWithPeer(C_2)$
$else$
  $UpdatePeer(C_2)$
$endif$
```

Equation 6.2: The threshold to pause synchronization with peers. The Synchronization occurs every reputation aggregation interval.

6.2.1 Distributed RepuScore Architecture and Algorithm

The architectural components of the distributed RepuScore are the RepuServer, RepuCollector which are similar to the components of the central RepuScore. However, in distributed RepuScore, there is no Central Authority. Figure 6.3 shows the different components of the distributed RepuScore algorithm.

A RepuCollector synchronizes with its peers to achieve a high-level of accuracy. The distributed RepuScore should achieve the baseline accuracy of the centralized RepuScore, but should be able to thwart the attacks that centralized RepuScore can face.

Reputation Computation

The RepuServer and RepuCollector compute reputation using Equation 4.2 and Equation 4.3 respectively. The RepuCollector reputations are shared among the peers using a distributed mechanism. Equation 6.1 shows the new RepuCollector algorithm for synchronization architecture. This algorithm is iterated a number of times till each node assume a steady state. Equation 6.2 shows the algorithm to compute the synchronization. The steady state in reputation is calculated when a system synchronizes with its peers. The steady-state is assumed to have been reached when the difference in the reputation between a previous time and the present time should be less than a threshold. This mechanism allows dynamic variation of the number of times a particular RepuCollector needs to synchronize with its peers.

The distributed RepuScore architecture faces Sybil Attacks similar to the Centralized RepuScore algorithm. The weighted RepuCollector algorithm above attempts to reduce the effect of Sybil domains by employing high importance to reputable peers than to low reputable peers.

6.2.2 Effect of Synchronization

Organizations can choose to perform unidirectional or bidirectional synchronization with the peers. Though bidirectional synchronization would converge faster, there might be security and privacy issues for bidirectional synchronization. For example, when a RepuCollector synchronizes with a peer, the peer trusts the requesting peer with its data. However, the data from the requesting peer might not be accurate enough for synchronization. Therefore, RepuCollectors should be allowed to perform both unidirectional and bidirectional synchronizations.

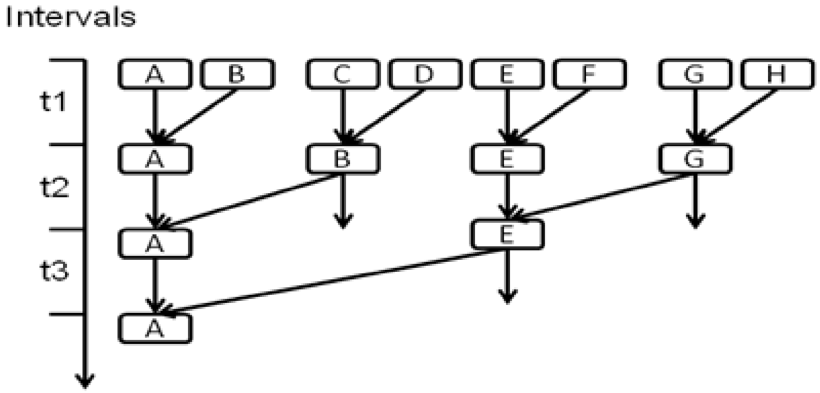


Figure 6.2: Effect of synchronization in distributed RepuScore.

Due to synchronization, a peer's local reputation is passed onto the senders through multiple iterations. Therefore, the sender reputation can be diluted multiple number of times before reaching a required organization. Figure 6.3 shows the effect of synchronizing. Reputations from node H is passed through the nodes G and E before reaching the node A. The reputation view of the node H is reduced by node G's reputation of node H, node E's reputation of node G and finally, node A's reputation of node E. It needs to be noted that: A→H ≠ A→E * E→G * G→H where → represents the synchronized value. Given number of RepuCollectors is n and the number of synchronizations at best case should be: log2 (n).

6.2.3 Evaluation of Distributed RepuScore

Distributed RepuScore based on the following criteria:

a) Synchronization Frequency: Synchronization frequency is the number of times distributed RepuScore needs to synchronize in a single day. The Synchronization frequency should be set to a value so that the cost of synchronization is minimized while computing the local reputation is highly accurate.

b) Effect of random peers vs. select set of peers: Distributed RepuScore can be designed to select random peers to synchronize their reputations or select a set of trusted peers to interact with. An evaluation should demonstrate which among the two provide higher accuracy.

c) Attacks on Distributed RepuScore: Distributed RepuScore needs to be evaluated to check how well it can fare against different attacks.

o Directed-attack against a single domain: a spamming domain can send directed spam against one domain while sending good emails to other organizations to maintain high reputation. Distributed RepuScore will be able to thwart directed attacks. Both the RepuScore algorithms could be evaluated to demonstrate how Directed-attacks can be thwarted.

o Sybil Attacks are where spammers create multiple identities that are used to gain disproportionate influence to thwart reputation management protocols. Experiments should be able to demonstrate that Distributed RepuScore can thwart Sybil attacks as best as RepuScore.

REFERENCES

Ahmed S.; F. Mithun. 2004. Word stemming to enhance spam filtering. In Proceedings of the First Conference on Email and Anti-Spam (CEAS).

Allman E. DomainKeys Identified Mail (DKIM): Introduction and Overview, 2005. www.mipassoc.org/ dkim/info/DKIM-Intro- Allman.html

Andreolini M.; M. Colajanni; F. Mazzoni; L. Messori. 2005. HoneySpam: Honeypots fighting spam at the source, In Proc. USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop, Cambridge.

Biswas S.; R. Morris. 2005. ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. Proceedings of ACM SIGCOMM ’05, Philadephia.

Brondmo H. P.; M. Olson; P. Boissonneault. 2003. Project Lumos: A Solutions Blueprint for Solving the Spam Problem by Establishing Volume Email Sender Accountability.

CAN-SPAM Act: Requirements for Commercial Emailers. http://www.ftc.gov/bcp/conline/pubs/buspubs/canspam.htm

Chirita P.; W. Nejdl; M. Schlosser; O. Scurtu. 2004. Personalized reputation management in P2P networks. Technical report, University of Hannover.

CipherTrust. 2006. TrustedSource: The Next-Generation Reputation System. White Paper.

Dewan P.; P. Dasgupta. 2004. Pride: peer-to-peer reputation infrastructure for decentralized environments. In Proceedings of the 13th international World Wide Web Conference on Alternate Track Papers & Posters.

Douceur J. R. 2002. The Sybil Attack. First international Workshop on Peer-To-Peer Systems.

Duan Z.; K. Gopalan; Y. Dong. 2005. Push vs. Pull: Implications of Protocol Design on Controlling Unwanted Traffic, In Proc. USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop, Cambridge.

Finnegan O. 2005. Email Deliverability Getting your Email into the inbox. http://www.ieinternet.com/mailwall/ Email_Deliverability_whitepaper.pdf

Gomes L. H.; C. Cazita; J.M. Almeida; V. Almeida; W. Meira. 2004. Characterizing spam traffic. In Proc. 4th ACM SIGCOMM Conference on Internet Measurement.

Goodmail Systems. Certified Email. www.goodmailsystems.com/certifiedmail.

Goodrich M.T.; R. Tamassia; D. Yao. 2005. Accredited DomainKeys: A Service Architecture for Improved Email Validation. Second Conference on Email Anti-Spam.

Gray A.; M. Haahr. 2004. Personalized, Collaborative spam filtering. In Proc. the First Conference on Email and Anti-Spam (CEAS).

Habeas Safe List, http://www.habeas.com/en-US/Senders/Safelist/.

Habeas SenderIndex, http://www.habeas.com/en-US/Receivers/SenderIndex.

Hardy I.R. 1996. The Evolution of ARPANET Email. Thesis, Department of History, University of California.

Hildebrandt R.; P. Koetter. 2003. The Book of Postfix.

IronPort. 2006. Study on Email Authentication Reveals Significant Adoption.

Jakobsson M.; S. Myers. 2006. Phishing and Countermeasures, Understanding the Increasing Problem of Electronic Identity Theft, Wiley.

Jordan S.; M. Blumberg; D. Cahill; R. Gingras. 2006. Accountable Email: Building on Authentication. Authentication Summit II.

Kamvar S. D.; M. T.Schlosser; H.Garcia-Molina. 2003. The EigenTrust Algorithm for Reputation Management in P2P Networks. In Proceedings of the Twelfth International World Wide Web Conference.

Kang B. B.; G. Singaraju; S. Jain. 2006. Privilege messaging: an authorization framework over email infrastructure. In Proceedings of the 20th Conference on Large installation System Administration.

Leiba B.; N. Borenstein. 2004. A multifaceted approach to spam reduction, In Proceedings of the First Conference on Email and Anti-Spam (CEAS).

Microsoft Corporation. 2004. Sender ID Framework – Executive Overview.

Milletary J. 2006. Technical Trends in Phishing Attacks. http://www.cert.org/archive/pdf/Phishing_trends.pdf

NACHA. 2004. Phishing losses total $500 million. Technical report, NACHA- The Electronic Payments Association.

Neustaedter C.; A.J. Bernheim Brush; Marc A. Smith; Danyel Fisher. 2005. The Social Network and Relationship Finder: Social Sorting for Email Triage. In Proc. Conference on Email and Anti-Spam (CEAS).

Papaioannou T. G.; G. D. Stamoulis. 2004. Effective use of reputation in peer-to-peer environments. In Proceedings of the 2004 IEEE international Symposium on Cluster Computing and the Grid.

Peterson P. 2006. SIDF and DKIM overview Scorecard. Authentication Summit II. http://www.aotalliance.org/summit_archive/pdfs/2_Summit_Scorecard_final.pdf.

Prakash V.V.; A. O'Donnell. 2005. Fighting spam with reputation systems. Queue 3, 9.

Price W. 2003. Inside PGP Key Reconstruction, A PGP corporation White paper.

Provos N. 2004. A virtual honeypot framework. In Proceedings of the 13th Conference on USENIX Security Symposium.

Ramachandran A.; N. Feamster; D. Dagon. 2006. Revealing botnet membership using DNSBL counter-intelligence. In Proceedings of the 2nd Conference on Steps To Reducing Unwanted Traffic on the internet - Volume 2 (San Jose, CA).

Realtime Blackhole List. 2002. Mail Abuse Prevention System LLC, California. http://www.mail-abuse.org/rbl/.

Rigoutsos I.; T. Huynh. 2004. Chung-Kwei: A pattern-discovery-based system for the automatic identification of unsolicited e-mail messages. In Proceedings of the First Conference on Email and Anti-Spam (CEAS).

Return Path. Sender Score Email Reputation Management, http://www.returnpath.com/delivery/senderscore.

Sahami M.; S. Dumais; D. Heckerman; E. Horvitz. 1998. A Bayesian approach to filtering junk e-mail, in: Proc. AAAI Workshop on Learning for Text Categorization.

Segal R.; J. Crawford; J. Kephart; B. Leiba. 2004. Spamguru: An enterprise anti-spam filtering system. In Proc. of the First Conference on Email and Anti-Spam (CEAS).

Sender Score Certified, Return Path Management, http://www.senderscorecertified.com.

Shmatikov V.; C. Talcott. 2003. Reputation-based trust management. In Workshop on Issues in the Theory of Security.

Singaraju G.; B. Kang. 2007. RepuScore: Collaborative Reputation Management Framework for Email Infrastructure, USENIX 21th Large Installation System Administration Conference.

Singaraju G.; J. Moss; B. Kang. 2008. Tracking Email Reputation for Authenticated Sender Identities, Fifth Conference on Email and Anti-Spam, 2008.

Srivatsa M.; L. Xiong; L. Liu. 2005. TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Networks. In 14th World Wide Web Conference.

Swamynathan G.; B. Y. Zhao; K. C. Almeroth. 2008. Exploring the feasibility of proactive reputations: Research Articles. Concurrent Computing.

Taylor B. 2006. Sender Reputation in a Large Webmail Service. Third Conference on Email and Anti-Spam (CEAS 2006).

Wikipedia. Challenge-response Authentication. http://en.wikipedia.org/wiki/ Challenge-response_authentication

Wong M. W. 2004. Sender Authentication: what to do, Technical Document, http://www.openspf.org/ whitepaper.pdf

Yahoo Inc. DomainKeys: Proving and Protecting Email Sender Identity. http://antispam.yahoo.com/domainkeys

Yu B.; M. P. Singh. 2002. An Evidential Model of Distributed Reputation Management. Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems.

Yu B.; M. P. Singh. 2003. Detecting Deception in Reputation Management. Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS), Melbourne, ACM Press.

Yu H.; M. Kaminsky; P. B. Gibbons; A. D. Flaxman. 2006. Defending against Sybil attacks via social networks. Proceedings of ACM SIGCOMM Conference.

Zimmermann P. 1995. The Official PGP User’s Guide. MIT Press, Cambridge.