Leveraging Email based Social Networks to Prevent Spam

Framework, System Design and Evaluation

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
"Doctor rerum naturalium"
der Georg-August-Universität Göttingen

vorgelegt von

Sufian Hameed

aus Sadiqabad, Pakistan

Göttingen 2012

Referent: Professor Dr. Xiaoming Fu
Korreferent: Professor Dr. Dieter Hogrefe

Tag der mündlichen Prüfung: 6th September 2012

Abstract

Spam is still an open problem from the network operator's perspective. The common state-of-the-art strategy is to place filters against spam at the recipient's edge. Although this strategy largely solves the spam problem from the user's perspective but the false positives/negatives may still exist. This strategy is also unable to prevent spam from traversing the Internet. Consequently, we now have around 260 billion spam messages sent across the Internet each day. Spam is therefore consuming large amounts of Internet bandwidth and is imposing non-negligible financial loss to network operators. Therefore it becomes imperative to mitigate spam much earlier than is typically done today.

The main part of this thesis proposes *LENS*, a novel, easily adaptable and scalable spam protection system that is incrementally deployable with low processing overheads. *LENS* leverages the recipient's social network to allow correspondence within the social network to directly pass to the mailbox of the recipient. *LENS* further mitigates spam beyond social circles and stops spam messages early on, instead of filtering these messages from user mailboxes or at the recipient's edge.

The key idea in *LENS* is to select legitimate and authentic users, called Gatekeepers (*GKs*), from outside the recipient's social circle and within predefined social distances. *LENS* utilizes the *GK* to generate a voucher, and new senders are required to obtain these vouchers (to communicate with the recipient) from a *GK* in their social neighborhood. Recipients recover from compromised GKs simply by selecting replacements and revoking vouchers. Unless a *GK* vouches for the emails of potential senders from outside the social circle of a particular recipient, those e-mails are prevented from transmission. In this way *LENS* drastically reduces the consumption of Internet bandwidth by spam.

The contributions of this thesis are the development of social network based spam mitigation framework (*LENS*), a system design based on this framework, the evaluation of the system by means of simulation and a prototype implementation. The evaluations show that with the help of hundreds of *GKs*, *LENS* can provide reliable email delivery from millions of poten-

tial users. *LENS* imposes Zero overhead for the common case of frequent and familiar senders, and remains lightweight for the general case. Using real email traces, the simulations show that *LENS* is effective in accepting all legitimate inbound emails. Our prototype implementation of *LENS* in Postfix/MailAvenger shows that *LENS* consumes up to 75% less CPU, 9% less memory and it is around 2-3 orders of magnitude faster in processing emails than traditional solutions like SpamAssassin.

Acknowledgments

I would like to express my gratitude to Prof. Dr. Xiaoming Fu for his supervision of this thesis, for his support and continuous guidance and for all the valuable insights and feedback that he provided. His efforts enabled me to develop scientifically and personally during my studies.

I would also like to thank Prof. Dr. Hogrefe, Prof. Dr. Grabowski, Prof. Dr. Wrögötter, Prof. Dr. Rieck and Prof. Dr. Yahyapour for participating as members of my thesis committee. I am deeply grateful for the countless discussions with my colleagues and peers. They provided interesting thoughts, constructive criticism, and a stimulating environment which made learning, researching, and working highly enjoyable. I would like to thank Dr. Pan Hui, Dr. Nishanth Sastry, Dr. Mayutan Arumaithurai, Dr. Florian Tegeler, Konglin Zhu, Lei Jao, and Jiachen Chen for being outstanding colleagues and for their valuable support.

Finally, I would like to thank my friends and family for their absolute and implicit support that helped me immensely during this journey.

Table of Contents

Appendices

List of Tables

List of Figures

List of Abbreviations

AS	Attestation Server
CA	Certification Authority
CommList	Community List
DKIM	Domain Key Identitied Mail
DNS	Domain Name System
E	Edge
F	Friend
FoF(s)	Friends of Friend(s)
G	Graph
GK(s)	Gatekeeper(s)
IMAP	Internet Message Access Protocol
LAN	Local Area Network
MDA(s)	Mail Delivery Agent(s)
MS(s)	Mail Server(s)
MTA(s)	Mail Transfer Agent(s)
MUA(s)	Mail User Agent(s)

MX	Mail Exchanger Record
OSN	Online Social Network
PGP	Pretty Good Privacy
PK(s)	Public Key(s)
PKG	Public Key Generator
PKList	Public Key List
POP	Post Office Protocol
R	Recipient
RE:	Reliable Email
S	Sender
SK(s)	Secret Key(s)
SMTP	Simple Mail Transfer Protocol
SPF	Sender Policy Framework
S/MIME	Secure/Multipurpose Internet Mail Extensions
TA	Trusted Authority
TCP	Transport Control Protocol
TR	Trust Ratings
UT	User Type

V Vertex

VM Virtual Machine

Chapter 1

Introduction

Simple Mail Transfer Protocol (*SMTP*) is an Internet standard that defines the transmission of electronic mail (email) between different hosts across the IP networks. *SMTP* was designed with an emphasis on simplicity. It has a simple addressing scheme with low implementation and administration cost. This has resulted in huge popularity and support of the protocol on many platforms by different vendors.

The original *SMTP* specification focused on designing simple, flexible and open email systems. It did not contain the facility to authenticate the senders which made it easy to hide information about where an email message is coming from and what it contains. This omission of the early days led to a system that fundamentally trusts the participants. This however, has turned out to be a huge mistake and malicious users have utilized this weakness to their advantage for sending spams. Unfortunately, extensive modification of *SMTP* or its complete replacement, is not practical, due to the network effects of the huge installation base of *SMTP* servers. Over the years spam has become such a nuisance, incurring huge financial and infrastructural cost, that by law spamming is considered a punishable crime in developed countries.

This thesis proposes a novel spam protection system that leverages social network of a recipient to extend the underlying trust infrastructure for mitigating unwanted/spam emails. The rest of the chapter discusses the motivation behind the proposed approach for spam mitigation and lists out the key contributions in the thesis.

1.1 Background and Motivation

Currently, it is just as easy to send an email to a stranger as it is to send it to friends. This has led to an immense abuse over the past decade in the form of unsolicited emails (spam). Despite the fact that researchers and practitioners have developed and deployed a broad variety of systems intended to prevent spam [8, 10, 11, 16, 40, 41, 50, 71, 75], spam messages keeps on increasing. In this cat-and-mouse game between e-mail spammers and

the anti-spam community, the spammers are still winning with spam emails largely outnumbering legitimate ones. Spam emails have now increased from 65% [15] in 2005 to 89.1% (262 billion spam mails/day) in 2010 [4]. There are primarily two main reason for the huge infestation of spam. First, there is a microscopic infrastructural cost to send spam. Anyone with a simple computer hooked up with the internet and a mailing list can easily start a career in e-mail marketing. Second, there is no ownership of spam. Spammers remain anonymous, hiding their identities and true origins, due to which it is not easy to take legal actions against malicious individuals.

Existing approaches to combat spam falls roughly into four broad categories:

- Content-based filtering
- Sender authentication approach
- Header-based approach
- Social network and trust based approach

Each of the above approaches has certain disadvantages which are discussed in Chapter 2. In short, the common state-of-the-art strategy used today is to filter spam from the user's inbox (i.e. recipient's edge). The problem with this approach is that it allows spam to traverse the network, and incur non-negligible cost to network operators in terms of bandwidth and infrastructure (spam is projected to cost $338 billion by 2013 [13]). On the other hand, content-based filtering [1, 10] which is one of the most widely adopted defense mechanism against spam has turned the spam problem into an email classification problem and results in unreliable email delivery due to false positives and false negatives. A false negative is when a spam email is classified as legitimate and placed in the inbox, whereas a false positive is the classification of a legitimate email as spam. False positive can result in delayed or non-access to an important email, thereby may potentially have a serious business implication [45].

There have been innumerable attempts to mitigate spam, including recent solutions that exploit trust embedded in social networks to create solutions without false positives [39, 59]. *RE:* [39] proposed that recipients can trust senders in their immediate social neighborhood, and used a zero false-positive mechanism for vetting emails sent by their friends or friends

of friends (*FoFs*)[1]. However, emails coming from outside of this circle still needs to be verified by noisy and unreliable spam filters. In contrast, Ostra [59] introduced a careful system of credits that allows anyone to send email to anyone else, as long as their balance of credits allows them to get a token. Unfortunately, for Ostra to be successful, the entire network would need to adopt the system.

So far, the solutions to block spam has done nothing more than being a minor inconvenience to spammers. The spam infestation is growing faster than the anti-spam community can keep up with and most anti-spam techniques so far have been like pesticides that do nothing more than create a new, resistant strain of bugs. Hence, there is a need for new mechanisms that impose zero cost on the legitimate users and make it hard for malicious spammer to adapt or bypass. Further, the solution should focus on processing spam at an earlier stage of transmission to save bandwidth and infrastructural costs.

1.2 Thesis Contributions

The aim of this thesis is to create a system that, like *RE:* [39] can be deployed individually by small groups of users, but allows for a reach greater than *FoFs*. The most important novelty of the scheme proposed in this thesis over *RE:* is that *LENS* demonstrates that it is *feasible* and *practical* to extend the reach of social network trust beyond *FoFs*. In order to accomplish this, a per-recipient ego-centric view of *the entire social network of email users* is created. Anyone who is within a recipient's community (i.e., a friend or *FoFs* in the context of this thesis can email the recipient directly. To enable legitimate senders who are farther away, *LENS* enlists a set of trusted users, called GateKeepers (*GKs*) at various hop counts from the recipient. These *GKs* are used to vouch for new senders in the immediate social circle of the *GKs* by issuing them un-forgeable vouchers. Essentially, the protocol requires a first-time sender's (senders from outside the social circle of a particular recipient) mail server (*MS*) to present a valid voucher *before* sending the payload. This enables the recipient's *MS* to quickly terminate an invalid connection, saving valuable resources both in the *MS* and in the network.

Although this solution is simple and straightforward, there are several advantages. First, graphs of social networks are known to be small

[1]Henceforth, the terms *community*, *social network*, *social circle* or *immediate social neighborhood* will be used to refer to the set of friends + *FoFs* of a user

worlds [72] and have small dominating sets [21, 31]. Thus, with only a modest number of *GKs*, each recipient is able to cover a large part of the legitimate social network, allowing most people to reach it. Secondly, a compromised or rogue node is able to spam only those recipients who have chosen it as a *GK*. Since the process of *GK* selection is performed independently by each recipient, the breakdown of individual *GKs* does not compromise the security of most recipients. The affected recipients can curtail their damage by rotating their *GKs*. Finally, the protocol requires a first-time sender's *MS* to present a valid voucher *before* transmitting the data (header and body of the message). This enables a recipient's *MS* to filter an invalid connection at the sender's edge, thereby saving valuable resources both in the recipient's *MS* and in the network. This feature becomes especially important when dealing with spams with large payloads such as viruses or other attachments.

The key contributions of this thesis are summarized as follows:

1. Unlike the existing social network based approaches, *LENS* is not limited to the social circle of an email user. *LENS* proposes a feasible and practical approach to extend spam prevention beyond a user's social circle, covering all the communication scenarios for legitimate inbound emails.

2. One major approach in fighting spam is to look for the spamming email addresses or IPs and block them. This approach is harder than it looks, since spammers regularly change their whereabouts to bypass IP and address based filters. This thesis explores a different path and instead of identifying spammers, *LENS* tries to identify legitimate users inorder to facilitate genuine emails from reliable *MSs*.

3. *LENS* is capable of filtering all the legitimate inbound emails at the *SMTP* time, before the actual transmission of email payload. This enables the recipient's *MS* to quickly terminate any invalid connection and prevent the transmission of spam across the network.

4. The thesis also proposes *iSATS*, a new crypto-based email sender authentication mechanism. *iSATS* is complementary to *LENS* and provides a reliable way to bind the identity of a legitimate sender to an email. On the other hand, it is hard for a spammer to adopt or bypass the system without getting noticed.

Finally, *LENS* is extensively evaluated using three large social networks (Flickr, Facebook and Google Buzz) and two traces of email transactions

(Uni-Kiel and Enron). *LENS* is able to receive *all* incoming emails in the email traces by using less than 0.06% of users as *GKs* per recipient. The analysis of social network traces demonstrate that a recipient in the *LENS* system needs just a few hundred *GKs* in order to receive reliable emails from a substantial portion of the entire social network. For the email traces around 80% to 92% of the selected *GKs* have less than 300 recipients. Thus, even if an attacker is able to compromise a *GK*, the affected number of recipients is limited. Our stress tests and micro benchmarks on the prototype implementation show that the overheads imposed by the additional processing are tolerably small. *LENS* is significantly less compute intensive (up to 75% less CPU and 9% less memory) than current solutions like SpamAssassin.

1.3 Thesis Overview

In the remainder of this thesis, the related work is discussed in Chapter 2. Later on, Chapter 3 introduces the overall architecture and system design of *LENS*. Chapter 4 illustrates prototype implementation of *LENS* and email processing based on this prototype. Chapter 5 introduces a new crypto-based email sender authentication scheme complementary to *LENS*. Chapter 6 details upon questions concerning the security of the system. Afterwards, in Chapter 7 an in-depth evaluation of *LENS* is performed to demonstrate the scalability, effectiveness and efficiency of the system. Finally, Chapter 8 concludes the thesis.

Chapter 2

Related Work

Spam can loosely be defined as *unwanted* and *unsolicited* emails that target large sections of the "legitimate" populace of email users. While many of the legitimate emails can be categorized as regular correspondence (and therefore considered solicited or wanted), the ability to send unsolicited emails is crucial for bootstrapping new communications.

It has been observed that spammers need to send a huge number of emails simply because the end goal of a large fraction of spam is direct marketing, and the conversion rate is extremely low [48]. An interesting recent suggestion is to attack spam by blocking the handful of payment gateways that are used when a spam victim ends up making a financial transaction [52]. However, payment gateways could also end up being a moving target.

It is possible to sidestep from the issue of identifying unwanted emails and simply curtail the sending rate of spammers. Proof-of-work and computational puzzles have been used as pricing mechanisms that create resource bottlenecks and limit sending rates [17, 32]. However, today's spammers control large botnets and computational power is unlikely to be the primary bottleneck. Computational puzzles requiring human interventions (e.g. *CAPTCHAs*) are outsourced [30].

Researchers and practitioners have developed and deployed a broad variety of approaches intended to prevent spam. These approaches roughly fall into four broad categories reviewed in the following sections of this chapter.

2.1 Content-Based Filtering

Content-based filtering uses heuristics and machine learning methods based on filters and keywords for spam recognition. It is the most popular spam protection technique and is widely available in most free and commercial implementations. SpamAssassin [10] and DSPAM [1] are two representative examples. Apart from spam protection, content-based filters are also helpful in mitigating other types of unwanted communication, such as blog spam [57] and network-based security attacks [51].

Unfortunately content-based filtering exhibits several problems which limits its usage. These problems include the intrinsic cost of initialization and continuous adaptation of the filters [42, 74], false positives (unwanted email classified as legitimate) and false negatives (legitimate email marked as spam). False negatives are more or less just an inconvenience. Whereas, false positives are more of a serious concern since important/urgent messages are marked as unwanted and thus may be delayed or not received at all [44]. Moreover, spammers and filter developers are engaged in a continuous arms race [42], because the cognitive and visual capabilities of humans allow spammers to encode their message in a way that can by-pass the filtering programs.

2.2 Email Sender Authentication

Email infrastructure was not designed to verify the authenticity of a sender address/identity. This weakness is greatly exploited by zombie networks or botnets to send spam/phishing messages with forged ***from:*** addresses. It is a well-known fact that the majority of spam messages today, 88.2% of the total spam according to some estimates [67], are sent by botnets using forged addresses.

Email sender authentication mechanisms enable receivers to automatically distinguish forgeries from authentic messages. This section explores several sender authentication protocols proposed over the years out of which Sender Policy Framework (*SPF*) and Domain Keys Identified Mail (*DKIM*) are the most adopted ones.

PGP and *S/MIME*

Pretty Good Privacy (*PGP*) [23] and *S/MIME* are both cryptographic approaches that sign the message body using public-key cryptography and append the signature in the body. In *PGP*, Keys are stored in end-user keyrings or in public key-servers. Key management uses a peer-to-peer web-of-trust architecture. Whereas in *S/MIME*, management follows a hierarchical model similar to SSL and keys are signed by a certificate authority.

Despite their presence for a long time, *PGP* and *S/MIME* are not widely used. Majority of mail user agents (*MUAs*) support *S/MIME* by default, still vast majority of email sent out is not signed. There are no published results that give a basis for the low adoption of *PGP* and *S/MIME*, but according to some experts [76] average users don't sign their outgoing email because they find it inconvenient to manage keys and enter pass phrases.

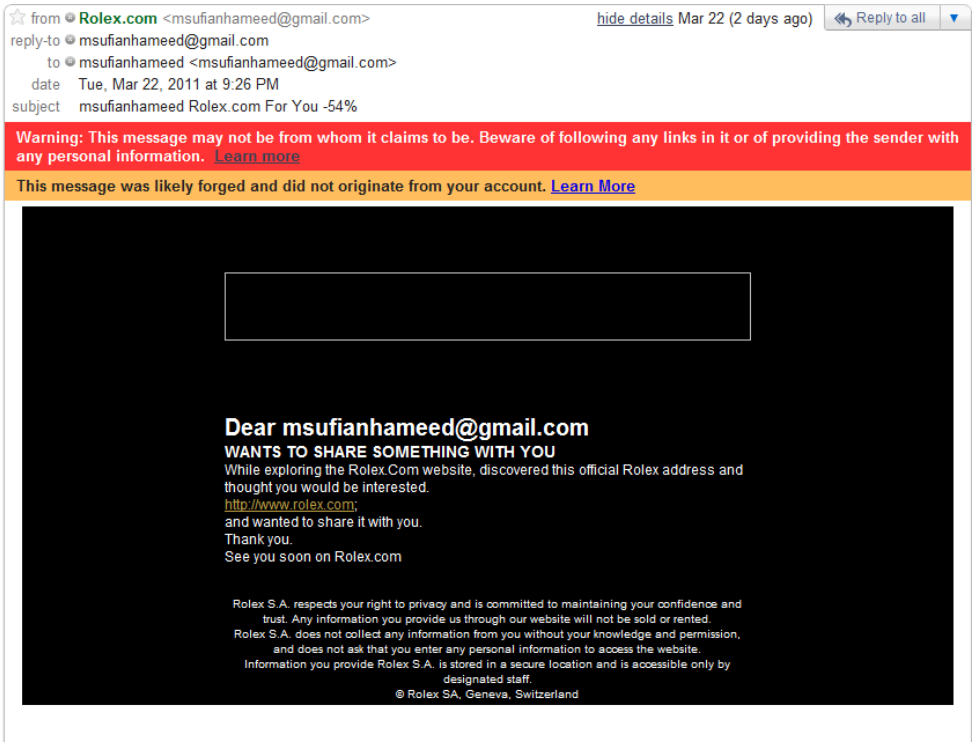

Figure 2.1: Example of an email with forged ***from:*** address.

Sender Policy Framework (*SPF*)

SPF [75] is an IP-based sender authentication scheme that operates on *SMTP* envelope (`MAIL FROM:`) to block forgeries at *SMTP* time. *SPF* allows the domain administrators to publish the IPs or range of IPs for their valid server(s) on *DNS* in simple text format referred to as *SPF* record. When the email exchange begins, the receiving side can query the *DNS* for sender's *SPF* record to validate if sender's IP is listed in the address range specified by the sender's domain.

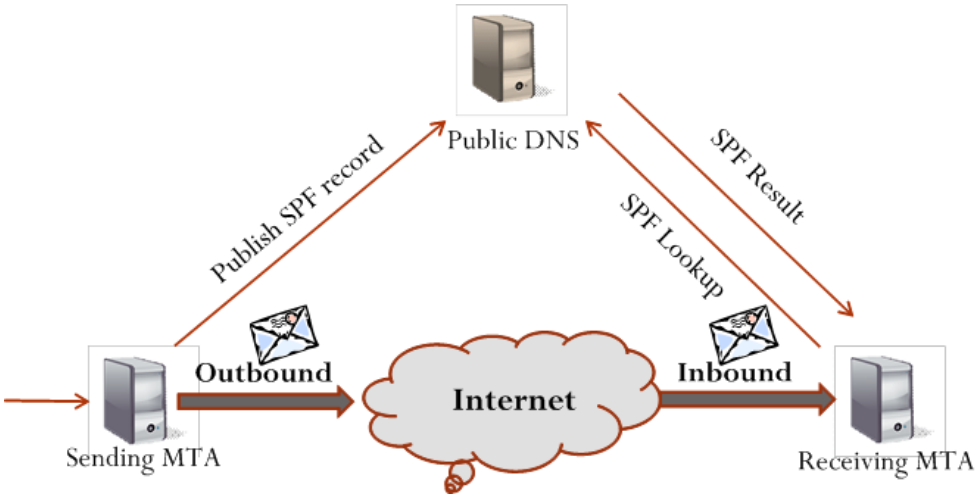

Figure 2.2: Publishing and lookups of *SPF* record via *DNS*.

According to [60], *SPF is the most adopted sender authentication scheme* and over 60% of the prominent domains have published their *SPF* record as of July 2011. However, due to its simplicity, *SPF* is also easily adopted by spammers. According to [60] 20+% of spamming domains have already adopted *SPF*. If the majority of spamming domains adopt *SPF* over time, *SPF* would become useless. [60] also showed that significant amount of spam is successfully authenticated by *SPF* and on the other hand around 5+% of legitimate messages can potentially fail *SPF* tests. In email forwarding, it is a common practice not to change the return path or `MAIL FROM:` message envelops and this is an Achilles' heel in IP-based *SPF* [76]. Unless the return path is edited during forwarding, the receiver will treat the message as forgery for not coming directly from its listed sender.

Domain Keys Identified Mail *(DKIM)*

DKIM [16] signs the email headers and body using public-key cryptography, and append the signature in a *DomainKey-Signature* header. The signature keys are bind to a domain name and the domain admins publish the public-key in the *DNS*. The receiver can query the *DNS* to extract the public-key of the sender and verify the signature. A successful verification implies that the message content was not forged during the transmission and the message is actually from the sender responsible for it.

Figure 2.3: Example of email header with *DKIM* signature and authentication results.

In *DKIM*, the signature can only be evaluated after the entire message content is received, thus, it is not possible to reject spam at earlier stages or during *SMTP* time. *DKIM* is prone to content munging and if the message content is altered during transit, *DKIM* will fail. *DKIM* cannot evaluate the trustworthiness of a sender and the spammers can also adopt it to sign their own messages. However, [68] shows that only 2% of spam received are authenticated by *DKIM*, which is significantly less than the spam authenticated by *SPF*.

Sender-ID

Sender-ID [56] can be interpreted as an extension to *SPF*. It relies on the *SPF* infrastructure and verifies the sender's IP address for authenticating the sender of an email. The basic difference between *SPF* and *Sender-ID* is that *SPF* verifies the envelope sender mentioned in the `MAIL FROM:` address, whereas, *Sender-ID* operates on purported responsible address (*PRA*) [55] of a message, which is the sender given in the header of a message. As with *SPF*, recipients can check if the purported sender's IP address matches the one that is published on the *DNS*. Since *Sender-ID* works on *PRA* in the message header, it cannot authenticate the message at *SMTP* time.

Microsoft holds the intellectual property rights to *Sender-ID* specification and Microsoft has published *Sender-ID* under a royalty free patent license agreement [28]. However, the license itself is a point of conflict in open source community and it is considered to be incompatible and contrary

to normal open Internet standards [29, 64]. Further, *Sender-ID* claims `MAIL FROM:` and *PRA* to be interchangeable at all times, but in reality this is not the case. In order to avoid compatibility issues, it would be necessary to publish records for *Sender-ID* separately.

Occam's Razor

Occam [36] is real-time challenge-based authentication protocol to validate the binding between individual domain names and legitimate mail sources for those domains. When an email arrives, the receiver simply sends a query to the sending domain identified in a message whether it actually sent the message. If the email is legitimate, the domain will acknowledge sending the message. On the other hand, if no acknowledgment is received from the identified sender then the receiver can classify the email as illegitimate. Operations of Occam protocol are illustrated in figure 2.4.

Being a challenge-based authentication system, Occam transfers the responsibility of email authentication to the sender on a per-domain basis. In order to join the system a spammer has to provide online servers capable of handling queries about any e-mail sent from that domain. This requirement increases the infrastructure cost on the spammer. Moreover, since the addresses of the servers are visible, the spammers are exposed during a spam campaign, thus becoming prime target for blacklisting.

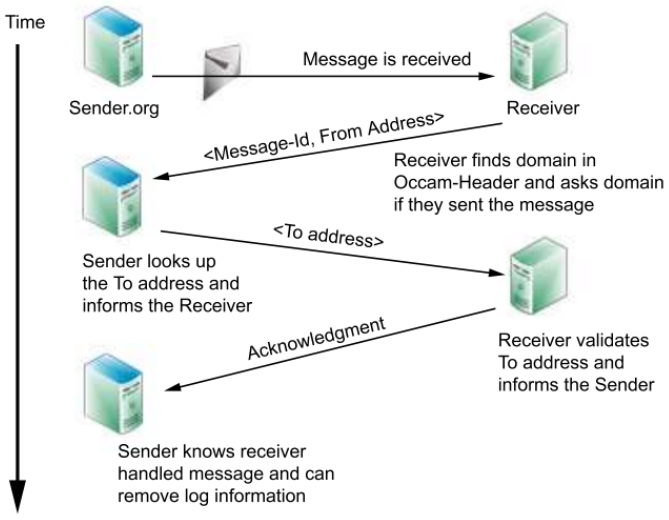

Figure 2.4: Occam's real-time challenge-based authentication (source [36]).

On the downside, spammers can abuse the challenge-based protocol as a reflector *DDoS* attack. A spammer can attack any domain by sending millions of messages claiming to be originated from that domain. As a result, receivers will then direct millions of validation requests to the domain identified in the Occam-header.

2.3 Header-Based Approach

Header-based approach examines the header of an email to detect spam. This approach can be categorized as white lists and black lists. *White lists* schemes collect a list of all the emails that are trusted to be non-spammers. Any email sourced from the addresses in the white lists is sent to the inbox. White listing is highly vulnerable to ***from:*** address forgery; therefore it must be used together with sender authentication schemes. *Blacklists* schemes, in contrasts stores the IP-addresses of all the spammers (email addresses are easily forgeable and are ineffective 95% of the time [26]) and refuse to accept emails from them. Manually generated lists have proved to be highly efficient but put a heavy maintenance burden on the email user.

2.4 Social Network and Trust Based Approaches

In *Social Network, Trust and Reputation Systems*, network users try to calculate the reliability and trustworthiness of other users based on their own experiences and that of others. This section reviews several recent techniques that use social networks and trust and reputation systems to combat spam.

Personal Email

Boykin and Roychowdhury have pioneered the idea of using social networks for message filtering [22]. Boykin proposed to create an individual user's trusted network of friends (social network) or personal email network solely from the sender and recipient information available in the email headers of all the email messages in that user's inbox.

Boykin analyzed the email header in the user's inbox to extract the list of recipient addresses, stored in the `To:` and `Cc:` fields. These addresses are then represented as nodes in the personal email network. The nodes (addresses) that appear in the same header i.e. that have communicated with the user are joined together via the edge. With the help of the structural

property of this personal email network, particularly the propensity of local clustering, email messages are classified as legitimate, spam or unknown based on clustering thresholds.

In the empirical studies of individual mail boxes, personal email network classified approximately 53% of all emails as spam or non-spam, with 100% accuracy i.e. with zero false positives. However, 47% emails were left unclassified by this network analysis tool and were passed onto existing filtering techniques. The proposed scheme has two very useful features. First, it is free from any user intervention or supervised training. Second, it is immune to false negatives i.e., non-spam being misclassified as spam.

RE: Reliable Email

RE: Reliable Email [39] introduce the idea that recipients can trust senders in their immediate social neighborhood, and gave a zero false-positive mechanism for vetting emails sent by their immediate social circle. *RE:* is an email acceptance system that uses whitelist of friends and friends of friends (*FoFs*) to increase the communication chance of only white list friends. *RE:* directly delivers the accepted messages to the recipient's inbox and passes the unidentified ones to the existing spam filters.

RE:'s main goal is to eradicate the unreliability introduced in emails by content-based filters and other spam fighting technologies which misclassify legitimate mail as spam. By using friends and *FoF* based whitelisting, *RE:* can accept almost 75% of the received emails and prevent up to 88% false positive by the existing spam filters. With the addition of *FoFs* protocol there is 10% increase in accepted emails compared to content-based filtering techniques.

Sender-based whitelisting is one of the simplest concepts used in email acceptance system. However, traditional whitelists have three issues limiting its usage. First, a recipient's whitelist cannot accept email from an unknown sender. Second, populating of whitelists requires manual effort. Finally, whitelists are vulnerable to forging of ***from:*** addresses.

In order to overcome these limitations, *RE:* automatically extends the set of senders whose mail is accepted by recipient's whitelists by explicitly examining the social network (friends and *FoFs*) among email users. Further, *RE:* appends an authentication token with each outgoing message to protect against ***from:*** address forgery. The *RE:* protocol is summarized as follows.

- Every user in RE: maintains a public key (PK) and secret key (SK)

pair. The user keeps the *SK* private and publishes his *PK* with his Attestation Server (*AS*).

- The users are allowed to issue an attestation that another user is a legitimate sender. If a user *A* attest a user *B* it will be written as *A* → *B*, which indicates that *A* is willing to have email from *B* directly forwarded to his mailbox. In other words, "User *A* trusts his friend *B* not to send him spam". An attestation contains the identities of the attester and the attestee along with the expiration date. Formally, an attestation is written as:

$$\text{A} \rightarrow \text{B} = (\ \text{Hash(A)}, \text{Hash(B)}, \text{start}, \text{duration})\ SK_A.$$

- All the domains participating in *RE:* are required to run an *AS*. The *AS* is responsible for storing the *PKs* of its users and the *PKs* of users attested to by its users. Further, it is also the responsibility of the *AS* to store the attestations both by and to the users of that domain.

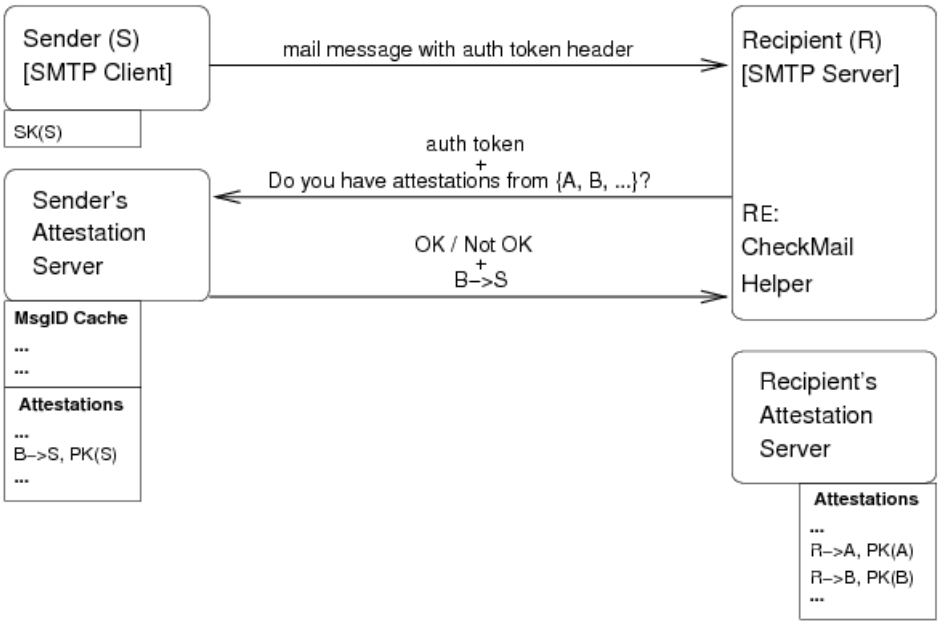

Figure 2.5: Email exchange and *FoF* attestation in *RE:* (source: [39])

- Figure 2.5 illustrates how an email is exchanged between a sender (*S*) and a receiver (*R*) running *RE:*. At the beginning, *S* composes a

message with authentication token and sends it to R. The authentication token is signed by the secret key of S and it is used as a defense mechanism against ***from:*** address forgery.

- On the receiving side, R can accept the email in one of two ways, depending on whether the sender is a friend or *FoFs*. For the validation of direct friendship, R examines the attestation list to see if it has attested for S. If $R \rightarrow S$ exists, *RE:* verifies the authentication token and accepts the message directly to the inbox.

- If S doesn't have a direct friendship with R, it will search for a *FoF* relationship. In other words, R will seek if S is a direct friend with any of his friends i.e. is there any x such that $R \rightarrow x$ and $x \rightarrow S$? To solve this mystery, R will send a *FoF* query to the *S's AS* with list of his direct friends. As a response the *S's AS* will send a list of attestations to the sender from those direct friends. Based on the presence of the attestation the *FoF* query is considered successful and *RE:* will accept the message to *R's* inbox.

In order to maintain the privacy of the users, R never sends its list of friends directly to S. R performs the *FoF* query using a private set matching protocol. This private matching protocol is realized with the help of special mathematical properties of particular public-key encryption schemes, such as Paillier [63] and a variant of ELGamal [34]. *RE:* has a couple of shortcomings listed below.

- Emails other than friends and *FoFs* still had to be filtered by noisy and unreliable spam filters. *LENS*, on the other hand extends the reliable delivery of emails beyond *FoFs* with the help of legitimate *GKs*.

- Cost of processing an email is too high and *RE:* employees a lot of structural overheads. Each user needs to maintain a pair of *PK* and *SK*, resulting in maintaining a system wide *PKI* (public key infrastructure) and *AS* for each domain. It would have been easier to use existing protocols like digitally signed emails, instead of going through all this architectural complexity.

- Discussion on the revocation of keys, authentication tokens and attestations is vague.

- *RE:* has an ill-defined security model. Freedman [37] has discussed the security short comings of *RE:* and proposed an efficient and stronger solution with similar cost.

It takes a village to stop spam

Seairth Jacobs [46] proposed a system where the users place more trust on their friends or social network. If an email is received from the social network it is directly accepted. On the other hand, if the email originates from outside the social network the recipient will ask his friends in the social network "do you know/trust this address"? The recipient will trust the email as safe if any of his friends trust the sender. If none of the recipient's friends know the sender they will ask the recipient to extend his query with their friends. The outline of the system proposed by Seairth can be summarized as follows.

The incoming emails fall into two categories, the email received from the known addresses i.e. friends and the other that are unknown. The user maintains a whitelist, containing the addresses of his friends or the users he directly trust. Any email coming from the addresses on the whitelist is directly accepted. The user also maintains a network file in the form of *FOAF* [14] that can be used by anyone to query the user. The *FOAF* file contains two different types of entries.

1. **Email address only.** This means that the user only knows the email address.
2. **Email Address and link to related person's *FOAF* file** (*FOAF* files are assumed to be shared on the web by each user). This indicates that the user not only knows the email address but also thinks that the person can be trusted and queried for the people he knows.

If an email is received from the email address listed in the whitelist it will be directly accepted. On the other hand if an email is received from an unknown address (e.g. abc@example.com) the user/system will one by one ask all the persons he knows (in the whitelist) to see if anyone is familiar with abc@example.com. If the answer is no the user will use the *FOAF* files of the friends and move to next hop/level of people to inquire about abc@example.com. The levels/hops to search and ask for opinion are left on the user's choice. Seairth also suggests that the weight of trust should be reduced as the number of hops increase. One way to do this is to increase the number of people who recognize the address. An example of this might be:

- 1-3 hops: at least one person must recognize the address.
- 4-5 hops: at least two people must recognize the address.

- 6-10 hops: at least four people must recognize the address.

The scalability of the system is a big question mark. For every unknown email received the same search intensive procedure will be applied which seems to be highly inefficient. Seairth has also proposed some optimization for the search operations, which includes limiting the number of queries a user can make to inquire about an unknown address and sorting the list of fiends based on the number of people they claim to know (highest number on the top).

SocialFilter

SocialFilter [66] proposes a collaborative spam mitigation system that uses social trust embedded in online social network and audits reports to assess the trustworthiness of spam reporter. Spam reports from the *SocialFilter* nodes are stored in a centralized repository that computes the trust values of the reports and identifies spammers based on IP addresses. Figure 2.6 depicts the high-level architecture of *SocialFilter*, comprising of following components.

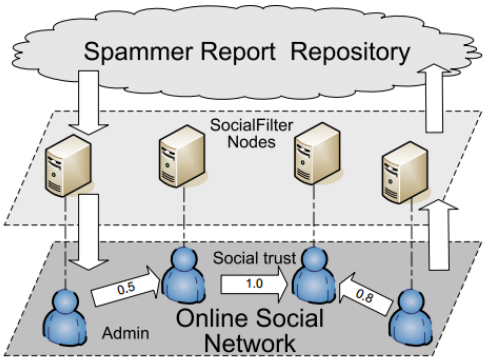

Figure 2.6: Architecture of *SocialFilter* (source [66]).

1. **Admins:** Human users that join a social network with a unique account and fulfill the responsibility to administer networked devices/networks.

2. ***SocialFilter* Nodes:** *SocialFilter* nodes are the spam reporters which are managed by human administrators (admins). These nodes are responsible for monitoring and reporting the behavior of the email

senders. Nodes maintained by trusted admins are expected to generate trustworthy spammer reports, while nodes managed by less knowledgeable admins are likely to generate unreliable reports.

3. **Spammer Reports:** These are the reports submitted by the *SocialFilter* nodes related to the status of email sender they observe. The reporting node also assigns a confidence level to its report. The weight of the report is calculated by the repository based on the reporter trust of a node, its identity uniqueness and its confidence level.

4. **Centralized Report Repository:** Centralized repository receives and stores spammer reports, and computes trust values.

In *SocialFilter* all the spammer reports concerning the spamming hosts are registered against their IP addresses. This may act as a limiting factor for *SocialFilter*, as spammers may use dynamic IPs. In general, IP blacklists are cumbersome to maintain and evadable [43].

SOAP

Figure 2.7 shows the structure of *SOAP*. *SOAP* integrates three new components to the existing Bayesian filter. Application of each of these components is briefly summarized below.

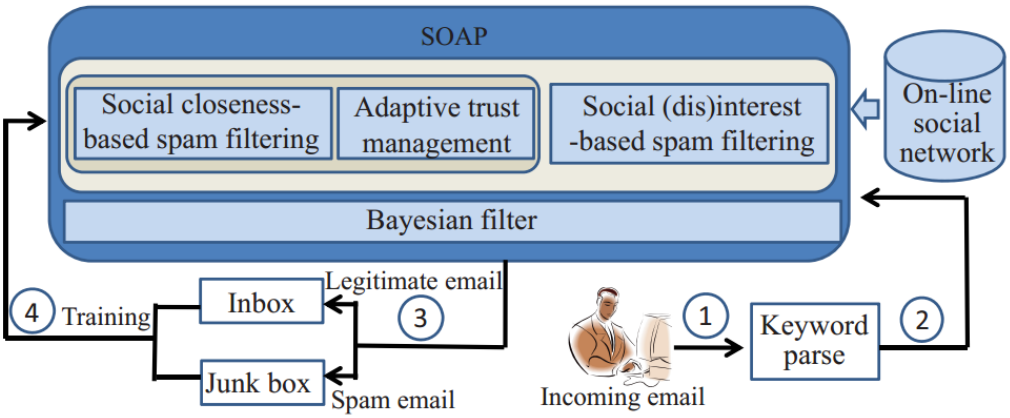

Figure 2.7: The structure of *SOAP* (source [53]).

1. **Social closeness-based spam filtering:** *SOAP* explores personal information in the social network to calculate the closeness between

nodes. Nodes with higher closeness are considered not to send spam, whereas emails from nodes having lower closeness are checked more strictly. Social closeness also helps *SOAP* against poison attacks.

2. **Social interest-based spam filtering:** *SOAP* gathers node's (dis)interests based on their social profiles and use it to enhance the accuracy of spam detection. This component helps *SOAP* realize personalized spam filtering.

3. **Adaptive trust management:** This component helps in preventing impersonation attacks. *SOAP* utilizes additive increase/multiplicative-decrease algorithm (*AIMD*) to regulate the trust values of nodes based on the social closeness values to block emails from low-trust or normal nodes due to impersonation by spammers.

There are several issues with *SOAP* that will limit its usage; this includes the intrinsic cost of initialization and continuous adaptation of social closeness (between sender and recipient) and social interests (of an individual) in the Bayesian filter. Moreover, the integration of online-social network is quite abstract in the description. Integrating an online-social network with an email network will raise a lot of privacy and ownership related concerns. *SOAP* will require complete access to the social graphs of the user, which is hard to achieve at the first place.

Ostra

Ostra [59] leverages the number of trust relationships a user has, for instance social links, to prevent unwanted communication in both email and content sharing systems such as YouTube. *Ostra* introduces a careful system of credits that allows anyone to send email to anyone else, as long as their balance of credits allows them to get a token.

Ostra connects senders and receivers via chain of pairwise trust relationships leveraging existing trust networks. With the help of a pairwise, link-based credit scheme *Ostra* imposes a cost on the sender if it generates an unwanted communication without requiring sender authentication or global identities. In *Ostra*, unwanted communication is classified based on the feedback from the receivers. Even in the absence of direct relationship between the sender and the ultimate recipient of the communication, *Ostra* guarantees that unwanted communication drains the originator's trust. This means that generation of continuous unwanted communication will eventually isolate the sender and will result in inability of the sender to communicate.

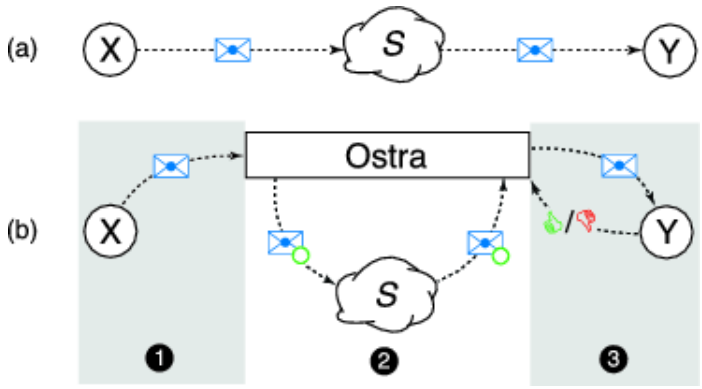

Figure 2.8: Diagram of (a) the original communication system S, and (b) the communication system with *Ostra* (source: [59]).

Ostra runs as a separate module along with the existing communication system. Figure 2.8 shows the three phases in which *Ostra* interacts with any given communication system S. Each phase is summarized below.

1. **Authorization:** In the first phase a sender generates and passes the communication to *Ostra*. *Ostra* then authorize this communication by issuing a token specific to the sender, recipient, and communication. Each user is allocated a credit balance on a per-user or per-link basis and this credit is used by *Ostra* to determine whether a token can be issued or not. In other words, if a sender has previously generated too much unwanted communication his credit would be negative. Due to which *Ostra* will refuse to issue a token and the communication would be rejected in the very initial stage.

2. **Transmission:** Once the token is authorized, *Ostra* attaches the token to the communication and uses the existing communication system to transmit it. On the receiving side, the communication is accepted only if a valid token is present. The communication is then passed on to the recipient.

3. **Classification:** In this phase the recipient classifies the communication as either wanted or unwanted, based on his/her preferences. This information is then passed on to *Ostra*, which makes this feedback available to the sender.

Scalability of this system is questionable specially when it has to maintain a per link credit scheme. Although a decentralized central tracker

component can be introduced in order to make the system more scalable, but such a scheme has not been implemented or evaluated. *Ostra* further leverages feedback to modify the weight of the edges in the social network dynamically for each message. This is a very big overhead on the system. Finally, for *Ostra* to be successful, the entire network would need to adopt the system.

MailRank

MailRank [26] also classifies and rank emails according to the address of email senders using a trust and reputation scheme which is similar to what Boykin *et al.* [22] does. The primary goal of *MailRank* is to allocate a rank to each email address known to the system and to use this rank to decide whether each email belongs to a spammer or not, and to accumulate a ranking among the filtered non-spam emails.

In order to compute relevant ratings for trust/reputation algorithms, it is essential to gather as many personal votes as possible. Furthermore, the maintenance of the system should require little or no effort at all to achieve a high acceptance. This means that the collection of personal ratings of each email address should require little or no manual user interactions and preferably it should be computed automatically. These goals are realized in *MailRank* by using already existing data inferred from the mutual communication between the users, i.e., who has exchanged emails with whom? There are three distinguish information sources that best served the purpose:

1. **User Address books:** If a user *A* has the addresses of users *B1, B2, ..., Bn* in its address book, then it is assumed that *A* trust those users.

2. `To:` **fields of outbound emails (i.e.,** `To:`**,** `Cc:` **and** `Bcc:`**):** If *A* sends emails to *B*, then *A* is assumed to trust *B*, or in other words *A* votes for *B*. This information is more accurate than address book entry, since the address book entry may be old or outdated. Furthermore, this data is also accessible via a light-weight email proxy deployed on any machine, whereas address books are private and require owner's permission in order to be accessed.

3. **Auto-whitelists:** Anti-spam tools (e.g., SpamAssassin) maintains a list of email addresses from which emails have been received recently, along with a score to determine if the address is spam or ham. All email addresses with a high score can be added to Auto-whitelists and can be viewed as trusted addresses.

TrustMail

TrustMail [40] presents an email scoring system that can identify good messages based on the reputation of the individuals who sent those messages. *TrustMail* extends the basic premises of whitelisting and social network based filtering. A user is allowed to assign "reputation" or "trust" rating to the people he knows i.e. can only assign ratings to people with whom he is connected. This results in large reputation network with thousands of interconnected users. TrustMail uses an individual user's personal view of the network and applies a recursive algorithm to surmise a reputation score for the sender of an email message.

The real beauty of the system is that legitimate emails from the sender previously unknown to the recipient can receive high scores due to interconnected social network. Thus, *TrustMail* helps in identifying good messages that might otherwise be blurry in the unwanted spam messages. On the negative side, TrustMail imposes a burden on the users to create an initial set of reputation. Furthermore, inferring the score of email sender for every single message consumes valuable resources of the email servers.

SNARE

SNARE (Spatiotemporal Network-level Automatic Reputation Engine) [43] is a sender reputation based email classification system. SNARE infers the reputation of an email sender using lightweight network-level features, without deep packet inspection i.e. without looking at the contents of a message.

SNARE examines lightweight network-level features; such as the distance in IP space to other email senders or the geographic distance between sender and receiver, to distinguish spammers from legitimate senders. In other words, *SNARE* classifies senders based on how they are communicating, rather than who the senders are (i.e., their IP addresses). *SNARE* focus on lightweight features, since they do not require deep packet inspection of large amount of emails from a single IP address and can be gathered without looking at the message's content.

Since *SNARE* is based on lightweight network-level features; it can scale better and operate on higher traffic rates. Furthermore, *SNARE* is capable of achieving accuracy comparable to existing static IP blacklists and classifies email senders as spammers or legitimate users with about a 70% detection rate having less than 0.3% false positives.

However, due to the lack of authentication and non-repudiation features in *SNARE* and other trust and reputation solutions make them vulnerable to

identity spoofing, false accusation and collusion attacks. In addition to this, these solutions consume extra resources of email servers on email reception and filtering. In contrast, *LENS* can reject unwanted email traffic during the *SMTP* transaction.

2.5 Conclusion

This chapter outlined the existing approaches to combat spam and discussed their advantages and disadvantages. There is no silver bullet against spam and spammers always find a way to get around whatever anti-spam filters are in place. In short, the common state-of-the-art strategy used today only filters spam at the recipient's edge, after it has traversed the network. This imposes a non-negligible cost on network operators in terms of bandwidth and infrastructure. On the other hand, content-based filtering which is one of the most widely adopted defense mechanism has turned the spam problem into a message classification problem. Recently, there have been some attempts to mitigate spam by exploiting trust embedded in social networks. These solutions propose a zero false-positive mechanism for vetting emails sent from within the social circle of a user. However, emails coming from outside this circle still need to be tested by noisy and unreliable spam filters. The remainder of this thesis will present *LENS*, a new spam prevention framework that extends the trust infrastructure to provide spam-free communication beyond the social network of a recipient. Furthermore, the proposed solution will filter spam during the *SMTP* transactions to save bandwidth and infrastructural costs.

Chapter 3

LENS Architecture and Design

Existing spam prevention solutions mainly focus on detection of spam and the spammers that are responsible for that spam. Every now and then new solutions are proposed to detect spam. However, the spammers also make improvisations, which have resulted in a continuous battle between the spammers and the anti-spam community. Instead of detecting spam, *LENS* mainly focuses on accepting legitimate emails from legitimate users. This is realized by the selection of legitimate users (community members in a close social circle of a user or socially distant trusted users called *GKs*) in a legitimate (*MS*).

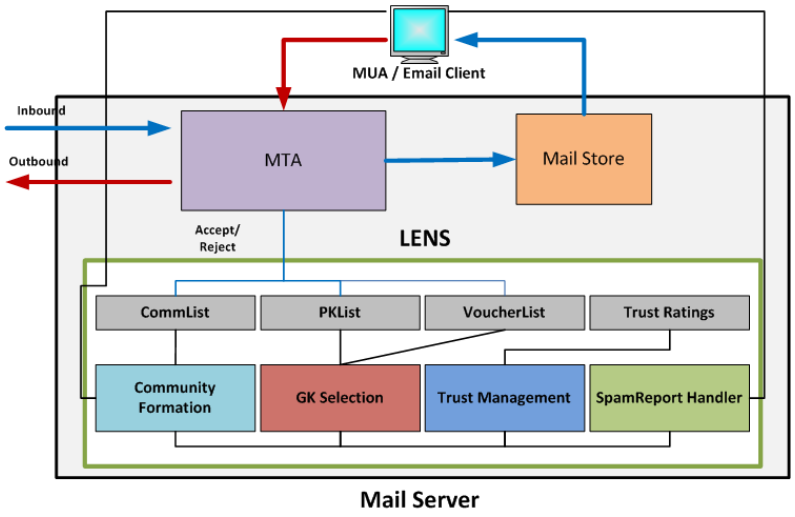

Figure 3.1: *LENS* Architecture

LENS comprises of four main components; 1) community formation, 2) trust management, 3) *GK* selection and 4) spam report handler, discussed in the following sections (figure 3.1). All these components run on a *MS*

along with the Mail Transfer Agent (*MTA*) and *SMTP* server. This chapter discusses the basic design goals of *LENS* followed by the discussion of *LENS* core components.

3.1 Design Goals

From the outset, we seek a solution that works well with the current, entrenched system. This means that the system follows several design principles:

1. **Simple and efficient design:** The simpler *LENS* is in terms of usability and design, the more effectively the system can be used and adopted. Since the motive is only to fight spam, there is no need for a very complex and robust cryptographic solution with substantially high infrastructure cost.

2. **Decentralized solution:** Usually, a centralized solution would be difficult to scale. Hence, in *LENS* every user works out individually to form his community and select his *GKs*.

3. **Knowledge of the network:** A user cannot obtain full information about the global properties of the whole social network, such as the network diameter, central nodes and node degree distribution. The *LENS* design does not rely on full knowledge of the network.

4. **Privacy of community list information:** Privacy of the community list of each user must always be protected against any external (i.e., outside the community) threats and they (i.e., lists) should not be exchanged freely at the time of community formation. Each individual node is not allowed to possess too much information about the network, which may induce privacy and security black-holes.

5. **Incremental deployability/backward compatibility:** *LENS* should be able to integrate easily into the current SMTP servers. Inevitably, when deployed, some users will adopt *LENS* before others. The deployment of *LENS* should not worsen the spam problem for those who have not adopted it. Until every user is familiar with *LENS*, it is better to run it complementarily with the existing spam filters (content-based filtering, blacklisting, whitelisting ...etc.). The users who will fully adopt *LENS* should get its full benefits; while others would profit according to the extent of their adoption. Nevertheless,

LENS should be backward compatible and should perfectly co-exist with the existing spam protection techniques.

3.2 Mail Server's Responsibilities

The *MS* is responsible for running *LENS* on behalf of the email users. Each *MS* may serve hundreds and thousands of email users depending on the size of the organization. It is assumed that each email user can explicitly control his community (friends and *FoFs*) and can give feedback by reporting spam emails. All the remaining functionality of *LENS* is handled transparently by the *MS*.

MSs running *LENS* are assumed to be legitimate with an extended validation certificate issued from a *Trusted Certification Authority (CA)* [35]. Most well-known webmail providers and websites tend to have certificates with extended validation of a domain's organization identity, so this requirement is not excessive. Furthermore, if a *CA* or its certificate is hacked, this will result in compromising all the communication to and from the website or *MS*. In such a scenarios spam will be of least concern. Discussion on such attacks and its defense is beyond the scope of this thesis. These certificates are used during server authentication (§ 3.7) to prove that the *MS* is legitimate. All the authentication requests associated with invalid certificates are ignored.

3.2.1 Legitimate MS and Botnets

In email networks there are mainly two ways to spam; a) human spamming, where a spammer creates an account (lets say on Gmail) and use it to send spam, b) using network of compromised and infected PCs (botnets) to send spams. Any good spamming model is based on cost free methods in terms of human and infrastructural resources. Majority of spam originates from botnets [36, 43] and according to [9, 12] around 80 to 85% of all email spam is produced by 6 to 10 botnets only.

It is hard, if not impossible, for bots or malicious users to reside in a valid and legitimate MS. The reason is that the addition of email users is strictly moderated in companies, private institutes and universities. Furthermore, all the major webmail providers run bot detectors against non-human automatic account creation. However, any one can create a large number of accounts on webmail providers like Gmail, Yahoo, Hotmail and GMX etc. Our previous assumption, that all the users within any certified *MS* are considered to be legitimate, might raise the question of human spamming;

since a human spammer will be able to create dummy accounts on webmail providers without any financial cost. In reality this is not the case due to the following reasons:

Webmail / Internet-service providers	**Email sending limit**
Gmail	500 recipients per message with web and 100 using email client software
Hotmail	100 recipients per day
Yahoo	100 emails or recipients per hour
Lycos	max 25 recipients per message and max 250 emails per day
AOL	100 recipients per message or 500 recipients per connection
Verizon	100 recipients per email and 500 emails (recipients) per hour
Comcast	1000 recipients per day
EarthLink	1000 recipients per day
Cablevision/Optimum (OOL)	50 recipients at one time
Road Runner	1,000 recipients per day per IP
AT&T Yahoo	100 recipients per email message
Charter	50 recipients / emails per hour

Table 3.1: Email sending limit by major webmail and Internet service providers.

- Creating and running a spamming account over Yahoo, Hotmail and Gmail requires human effort and all of this will incur cost which is inhibitive to the spamming model.
- Almost all the webmail and Internet service providers impose an email sending limit. Exceeding the limit, results in blocking of an email account for a certain amount of time [2]. Table 3.1 lists the email sending limit of some of the major Internet service and webmail providers. Apart from imposing limits on sending emails, webmail providers also block email accounts for a certain time if the email contains a large number of non-existent or broken addresses that bounce back on failed delivery.
- When a *MS* is certified (just like any web-server) by a *Trusted Certification Authority* it shows that the *MS* belongs to a legitimate owner or at least the identity of the *MS's* owner becomes visible, making it

easy to take legal action against the owner or blacklist the *MS*. Investing in infrastructure or becoming visible has never been a choice for the spammer.

Scale of human spamming is not significant compared to botnets. Therefore, studying the impact of human spamming is not a focus of this thesis, and it will be part of our future work. *LENS* addresses the problem of non-legitimate/human-spammer node by maintaining a trust rating (see § 3.4) for each user to ensure its legitimacy and differentiate between legitimate and illegitimate users.

3.3 Community Formation

Many of our important decisions in daily life are based on the information provided by our network of friends. Therefore, the reliability of our decisions depends on the trustworthiness of our social network.

A number of studies about social networks have presented measures of the closeness of a community. They have shown that these measures can be used to distinguish empirically observed social networks from non-social networks [33, 61, 62]. The most distinctive property of a social network is its tendency to form clusters. For example, if A knows B and C i.e. B and C are connected through A, then the likelihood of B knowing C is considerably higher than a random network with a similar degree of distribution.

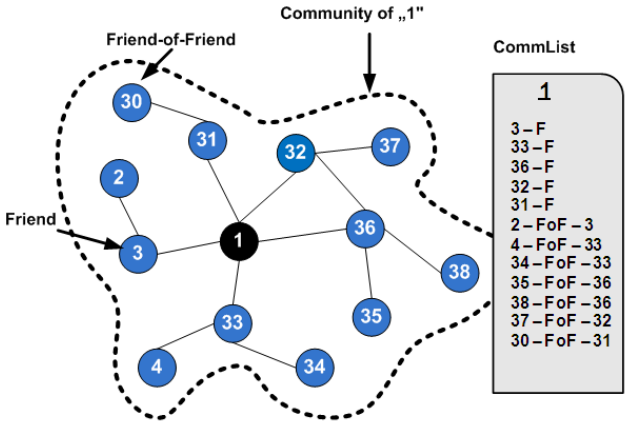

Figure 3.2: Community structure (friends, FoFs) of user **"1"**.

In *LENS*, the social components of a community consist of two levels,

namely friends of the users and his friends-of-friends (*FoFs*). Adding a friend roughly corresponds to the notion that "User A trusts his friend B not to send him spam and vice versa". Triadic closure is supported by adding FoFs into the community.

Figure 3.2 depicts the community structure for a user. Users can receive *all the messages from his community directly into his inbox*. The formation of a social community is a simple two-step process.

1. **Adding Friends:** The first step starts with the initiation of a friend request. Anyone can request anyone else to become his/her friend. Addition of a friend is the very basic yet extremely crucial step in *LENS*. It is assumed that only two users having a mutual trust on each other will enter into a friend relationship (like in MSN, Skype or Facebook). System security and defense against attacks from malicious users depends on the fact that friend's relationship is always formed between two legitimate users having proven record of social interaction. Once the two users add each other as friends, an entry is made in *CommList* (discussed below) with the user ID and label "F".

2. **Adding FoF:** The idea of *FoF* addition is that there will be no exchange of friend lists among the friends. Instead a user can suggest his mutually exclusive friends to add each other into their communities as *FoF*. For instance (figure 3.3), user "2" has two mutually exclusive friends so "2" will suggest both "3" and "1" to join the FoF relationship. If both "1" and "3" accept the suggestion they will add each other into their communities. Once two users add each other as FoF, an entry is made in their *CommList* with the ID of the added user, label "FoF" and the ID of the referring mutual friend.

By the end of Step 2, all the users will have a community structure with friends and *FoFs*. All the communities consist of only 2 levels of social components which are considerably close. During the entire process of community formation, only local information of direct neighbors is used and the process is carried out in a decentralized manner at each individual user level. Furthermore there is no exchange of friend lists among the users without consensus so that the privacy of each user can be protected. By design, community formation is a selective process and involves certain human involvement to prevent any unnecessary addition in communities and preserve high level of privacy.

Figure 3.4 shows a cross platform and email client independent community formation client which has been developed in *PyQT* to handle the

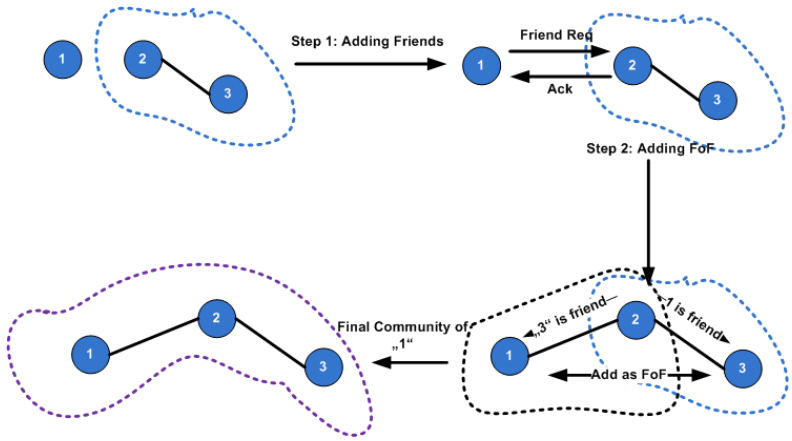

Figure 3.3: Community formation of user **“1”**.

process of community formation. The community formation client provides the functionality to add friends, view community, view pending requests, accept/reject the request received and send suggestions to mutual friends.

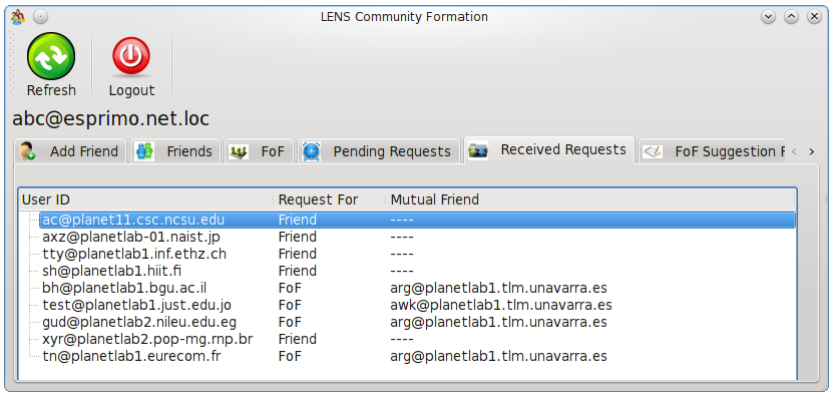

Figure 3.4: Cross platform community formation client in PyQT.

Community List (*CommList*)

CommList is a primary data structure that contains the records of the community members for each user in the system. This list is kept as a plain text file on the *MS* and maintains entries of the community users, either as friend or *FoFs* (see figure 3.2). *CommList* can only be modified by the

respective user and it is also visible (accessible) to the *MS* that host the user and his *CommList*. The *MSs* are authorized to query the *CommList* of a recipient during the *GK* selection process.

3.4 Trust Management

In order to ensure that illegitimate users are not selected as *GKs*, *LENS* maintains a system wide trust rating (*TR*) for each email user on the *MS*. The main goal of *TR* is to assign a rating to each user known to the *MS* and to use these ratings to decide whether the user is legitimate or not.

Based on the *MS's TR* there are four different user types (*UT*) i.e. *legitimate (LU), trusted (TU), new (NU)* and *illegitimate (IU)*. *LU* are non-spammers with a clean track record of spam-free communication and *TR* $>= L$ (trust rating threshold of legitimate user). *TU* as the name suggests are the users trusted by the *MS* to be legitimate/non-spammers. They have *TR* $>= L$ and the *MS* has verified the uniqueness of their identities. *NU* are newly registered users with the *MS* with $0 <= TR < L$. *IU* are the identified spammers with negative *TR*.

Assignment of *TR* falls in two main categories as follows.

Direct *TR*

Direct *TR* is the manual assignment of *TR* to a user by the admin(s) of the *MS*. Direct *TR* have priority over other methods of *TR* and it overrides the existing values of *TR*. Direct *TR* is more practical for small-medium size, strictly moderated *MSs* of companies, private institutes and universities, where the admins have direct trust on the users (for example users like Uni professor or company's GM are trusted not to be spammer). For big web-mail providers and ISPs the *TR* of each user should be computed automatically.

Automated *TR*

Automated *TR* is based on user voting. Analysis of spamming nodes shows that they exhibit one way communication i.e. they are always on the sending end [22, 26]. If *A* sends email to *B*, then it can be regarded as trusting *B*, or voting for *B* [26]. Trust Management component of *LENS* can be applied at the SMTP envelop (`MAIL FROM: or RCPT TO:`), where it can compute *TR* according to the algorithm summarized below. The *TRs* are maintained by

the *MS* for each of its user.

```
Algorithm 1:  Automated TR

OFFLINE

# Domain = D, UserType = UT, TrustRating = TR, Distinct Domains = DD

1: # infer votes from existing data

2: For Each inbound email from sender i to recipient j
3:   if (email == Legitimate)
4:      if (j has no votes from i)
5:         TRj +=1
6:      if (j has no votes from Di)
7:         DDj +=1
8:
9:   if (TRj >= L && DDj >= D)# reach votes threshold for Distinct Users & Domains
10:       UT = Legitimate

12:# indentity uniqueness test independent of voting
13: For Each  j with (UT == Legitimate)
14:     status = verify_identity_uniqueness(j)
15:     if (status == True)
16:         UT = Trusted

ONLINE

# Processing live email data after processing existing data
1: For Each inbound email from sender i to receiver j
2:    run step 3 to 16 of Initialization block
```

The automatic *TRs* are computed in two modes i.e. offline and online. The only difference between these two modes is the stream of data they operate on, other than that there is no difference in the functionality. In the offline mode the TRs are computed by mining the existing server logs of the email communication. Whereas, the online mode operates on the live email data streams to calculate the TRs. In order to gain higher *TR* the users are required to gather votes. In other words, the *TR* of a recipient increases on the reception of legitimate inbound emails.

The automated *TR* algorithm observes each inbound email from the sender. On the reception of a *legitimate email* (legitimacy of the vote is dependent on the legitimacy of the email) the TR of the recipient will be increased by one. Multiple emails from the same sender will have no effect on a recipient's *TR*. A sender can only be counted once for any particular recipient. When the recipient will receive legitimate email from L distinct senders belonging to D distinct domains, the UT of the recipient will be upgraded to LU (*legitimate*).

After becoming LU, a user has to pass the identity uniqueness verification in order to become TU (*trusted*). During this verification the user is binded

with his unique identity such as his mobile number using challenge response authentication. Facebook also verifies the uniqueness of user accounts by binding them with unique mobile numbers. A potential attack on trust ranking has been discussed in § 6 along with the means to neutralize it.

3.5 *GK* Selection

GK selection is a user transparent process carried out by the *GK Selection* component of the *MS* running *LENS*. This means that the user being selected as *GK* is also not aware, and has no control, of it being a *GK*. During the selection process, *LENS* selects good actors (legitimate and trusted users with good reputation) as *GK* and transparently uses them as a means to vouch users outside the community of the recipient R for communication. To maintain a reliable trust structure, a *GK* is only authorized to vouch for the users in its own community. On successful completion of the selection procedure, both the *(R)* and his *GK* receive their respective keys. *GK's* key is used (by *GK's MS*) to issue vouchers to the *GK's* community members so that they can communicate with R. On the receiving end, *R's* key is used to verify the vouchers. In order to keep *LENS* effective and scalable, the goal of GK selection process is to select **minimum number of GKs** for **maximum coverage**.

Let us consider all the email users as a connected network and visualize it as a graph $G = (V, E)$, with email users as vertices (V) and their relationships (i.e. friends) as edges (E). For every recipient node in G, a subset S' of V (i.e. *GK* nodes) is required such that nearly every vertex not in the social community of the recipient node lies within at least one of the communities of the member of S', and the size of S' is desired to be as small as possible.

The total email users today are more than 1.4 billion [3]. Finding a smallest subset of S' with maximum coverage raises the scalability question. This is also similar to the minimum dominating set problem [38], which is a classical NP-complete problem in computational complexity theory. The only difference is that a GK is connected to its community instead of direct neighbors.

In this thesis the classical dominating set or distributed dominating set approximation is not used to select the GKs for two main reasons. First, a common set of nodes to serve as GKs for the whole population are not preferred. The reason is that these common GKs will have too much in-

formation about everyone in the network and would become privacy and security weak points. The second reason is that a further consideration of the communication patterns revealed that one cannot expect everyone on the planet to communicate randomly with each other. The probability actually decreases with an increase in the social distance. Instead of working on a global provisioning of optimal GK for the entire email network, this section will discuss and present a scalable approximation.

One of the design constraints of *LENS* is that *a user cannot obtain information about the global properties of the social network*. The best approach here would be to restrict a user to his personal community information. The GK selection procedure of *LENS* consists of the following three stages, covering all the communication scenarios for legitimate emails.

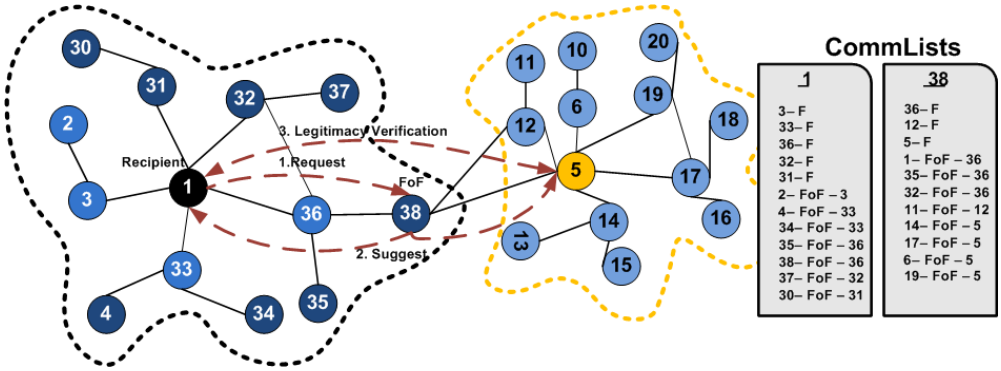

Figure 3.5: *GK* selection in adjacent communities.

3.5.1 Stage 1 - *GK* selection in adjacent communities:

The *GK* selection process for any recipient *(R)* starts with its adjacent communities as described below.

1. **Request:** The *MS* will use *R's FoF* (boundary user) to find the locally optimal *GKs* (figure 3.5). *R's MS* will simply request all *FoFs* of the community to send their suggestions for good *GKs*.

2. **Suggestion:** The *MS* of the *FoF* will suggest a potential *GK* from *FoF's* friends by selecting the user with the largest number of friends (outside *R's* community). The *MS* of the *FoF* will also inform the potential *GK* about R. For instance in figure 3.5, which depicts the selection of *GKs* by a *FoF* of *R*, user "38" suggests "5" as the locally

optimal *GK* to "1" instead of "12", since "5" is the friend with biggest community (and who is outside the community of "1"). *R* will choose the set of *GKs* that provide the best coverage. At the end of Stage 1, users within 5 hops of the *R* can send emails to it.

3. **Verification of Legitimacy:** This is the last and most important step of *GK* selection process. This step ensures that the GK is legitimate (details of this are covered in § 3.7). As a result of this step, a *RSA* based key pair (*PK* and *SK*) is generated for the *GK*. *PK* is shared with the *R* and the *SK* is use to issue vouchers to all the community members of the *GK*. These members will use the issued vouchers for communicating with the *R* (see figure 3.6).

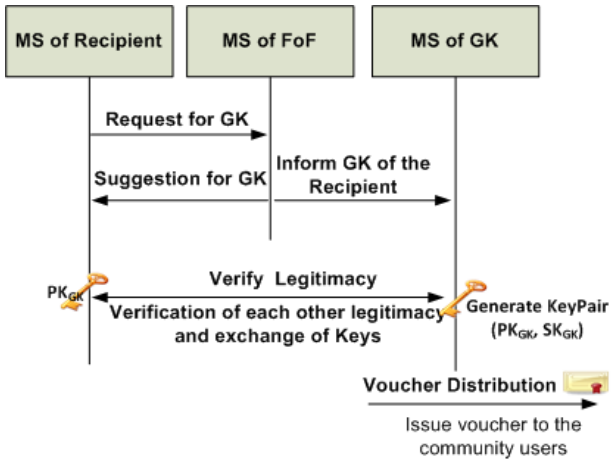

Figure 3.6: GK verification and voucher distribution.

After the stage 1, all the users within a social radius (level or hops) of 5 would be able to send emails to the recipient with an assurance of being free from spam. According to the small world property of social network, any two users can be connected with a small number of hops. This phenomenon is also popularly known as six-degree of separations [69], indicating that everyone is on average six steps away in social networks. According to latest studies on Facebook social graphs the degree of separation between any two Facebook users is even less than 5 hops [18, 70]. This suggests that if the email network exhibits a social network behavior, the recipient would be highly reachable. Nevertheless, distant users having a social distance greater than 5 are covered in stage 2 of the *GK* selection process. *PKList*

and VoucherList (discussed below) are maintained at each user's *MS* to store the *PKs* and vouchers. To communicate with each recipient outside of its social circle, a user would have to maintain a voucher for that recipient. Similarly, a GK issues a single voucher for all the recipients outside of his social circle for whom he is vouching.

Public-key List (*PKList*)

Public-key list or the *PKList* is maintained for the recipients and enlists the authorized *GKs* for the *R* along with its *PKs*. Any single entry in *PKList* contains the public-key and the identity of the selected *GK* $(PK, GKID)$. PKList can only be accessed and modified by the *R's MS*. Even the *Rs* have no access or control on the *PKLists*. This is because by design, the *GK* selection is a system process completely transparent to the users. If, for any reason, the *GK* is revoked, its corresponding entry is also deleted from the *PKList*.

VoucherList

As a result of *GK* selection a RSA-based key pair (*PK, SK*) is generated for the *GK*. The *MS* of the *GK* use the *SK* to issue voucher $(((User_{ID})_{hash})_{Sign-SK})$ to all the members of the *GK's* community. These vouchers are added to the *VoucherList* of the users (GK's community members), along with the *IDs* of the *R* and the *GK* $(< R_{ID1}, R_{ID2}, ... >, GK_{ID}, Voucher[((User_{ID})_{hash})_{Sign-SK}])$, to use for communicating with the *R*. A single voucher issued by the *GK*, to each of his community user, will work for all the *Rs* for which the *GK* is authorized to vouch.

3.5.2 Stage 2 - *GK* selection beyond adjacent communities:

After stage 1, the *GK* selection procedure of *LENS* can be easily extended to select *GK* in distant (beyond adjacent) communities. *R's MS* will send a request to the selected *GK's MSs* to help them look for *GKs* from their adjacent communities. As a result of this request, the *GKs* will use their *FoFs* to find new locally optimal *GKs* and send their suggestions back to *R*. (see figure 3.7). This process creates the second outer ring and extends reachability of *R* to users which are up to 8 hops away. This can clearly be extended to create additional rings of trust, but our implementation stops at Stage 2. While selecting *GKs* at Stage 2, it is important to ensure that the users selected as *GK* are not included in an inner ring. When the *CommList* is available directly, this is straightforward. The *GK* selection for

higher levels must also consider the small world property of social networks [69], in order to *avoid random walks* on the social graph.

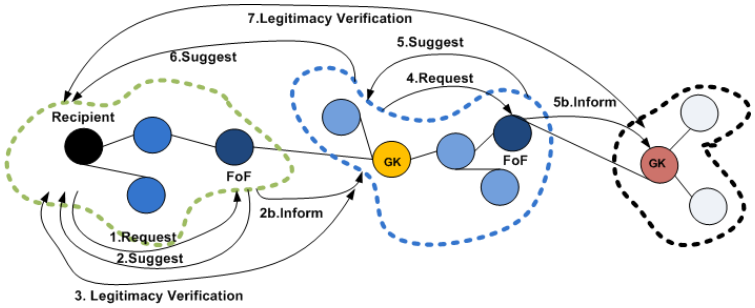

Figure 3.7: *GK* selection beyond adjacent communities.

3.5.3 Stage 3 - *GK* selection for new communication:

One of the reasons that email is so great is because anyone can contact anyone else. People get legitimate emails from new people everyday. *LENS* provides spam free email communication to distant and new users as follows. Instead of extending *GK* selection to the entire network, *LENS* restricts it only to the social levels covered in stage 1 and 2. If a user wants to send an email to R for the first time, which is not only outside its community but there is also no *GK* for R within its community, *LENS* will perform the following two steps.

1. **Announcement:** announce the sender (S) to R that wants to communicate and start the legitimacy verification process.

2. **Verification of Legitimacy:** start the legitimacy verification process (see § 3.7 for details) to prove that S is not a spammer. As a result of this process, R will add S as its GK. *S's MS* will further issue vouchers to his entire community and they will be able to use these vouchers to communicate with R.

This process is only performed once at the start of a new communication. After the S is verified as a *GK*, not only the sender but his entire community can send email to the R. Therefore, instead of having a *GK* for the entire network in *LENS* a *GK* can be selected on the fly after stage 2. The results in § 7.2 show that the need for stage 3 *GK* selection is quite infrequent.

Further, the results in § 7.3 shows that stage 3 only adds a negligible delay of 0.5 to 1.5 secs in email transmission.

RE: received 85% of the emails correctly by utilizing just its social network (friends and *FoFs*) [39]. With the *GKs*, *LENS* enhances its reliable and spam free delivery of emails beyond social network. Success of *LENS* depends on the successful formation of social communities and continuous extension of GK selection between the email users. Users having a larger social community would benefit more from *LENS* as compared to isolated and less socially connected users.

3.6 Spam Report Handler

Spam reports are handled by the *Spam report handler*. In *LENS*, by design, only reports from *trusted (TU)* are weighted. This design decision has been taken to prevent spammers from falsely reporting non-spammers as spammers. When a user will receive a spam, he will report to his *MS* that the sender is a spammer. Upon receiving a spam report, the *spam report handler* will register the report against the *TR* of the reporter. Once the handler receives the *RT* (report threshold) reports from distinct trusted users, it assigns a negative *TR* to the spammer. Furthermore, if the spammer is not a local user on the *MS*, the handler will add the offending user to the revocation list thereby preventing further spamming. Handling of spam report is summarized in Algorithm 2.

```
Algorithm 2:  Spam Report Handling

# Domain = D, UserType = UT, TrustRating = TR
# SRij = # of SpamReports by user i of user j
# Ij = # unique reports against j
# RT = threshold
ForEach SpamReport of Spammer s by Reporter r

   if(UTr == trusted && SRrj ==0)
      Is++
   if (Is >= RT)
      if (s = localuser)
         TRs = negative
         UTs = illegitimate
         delete s from CommList, VoucherList, PKList
      else:
         Add s to Revocation list as IU
```

After malicious users (local or remote) are identified and marked as *illegitimate*, their associated entries in *CommList, PKList* and *VoucherList* are also terminated from the *MS*. This means that if a *GK* is malicious, its

associated entries from the *PKList* of the recipients and *VoucherList* of the community members will be removed.

3.7 Legitimacy verification in *GK* Selection

Legitimacy verification is one of the most significant part of *GK* selection process. This protocol ensures that the *R* can trust the *GK* to be legitimate. With this protocol a *RSA* based key-pair (*PK* and *SK*) is generated for the *GK*. *PK* is shared with the *R's MS* and the *GK's MS* uses the *SK* to issue vouchers to his community members as a vouching mechanism to send emails to the *R*.

Based on the trust rating - *TR* (please refer to § 3.4 for details on trust management) there are four different user types (*UT)* i.e. *legitimate (LU), trusted (TU), new (NU)* and *illegitimate (IU)*. The following sections explore two variations of the protocol based on the difference of the locations of the *R* and its *GK*.

3.7.1 Recipient and *GK* on different *MS*

If the *R* and the *GK* belong to different *MSs*, legitimacy is verified in two steps. First, ***server authentication*** is carried out based on extended validated certificate issued by a *trusted certification authority* to verify that both the *R's* and the *GK's* servers are legitimate and un-tempered. Second, the *TR* of the user at his *MS* is verified to ensure that the user is not illegitimate/spammer. Following is the description of our legitimacy verification protocol. In *LENS*, by design, *GK's MS* will always initiate the protocol.

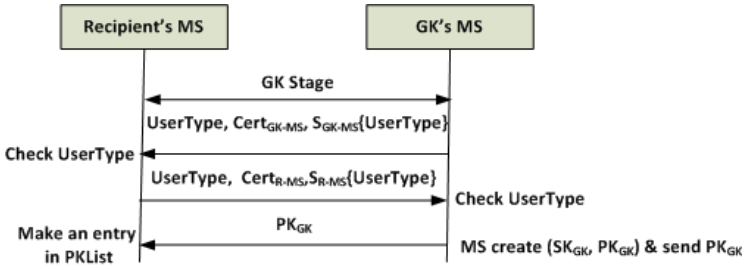

Figure 3.8: Legitimacy verification in *GK* Selection.

- *R's MS* and its *GK's MS* agrees on the stage of *GK* selection.
- *GK's MS* picks the *UT* of the *GK* (based on the *TR*), signs the *UT* using his private signature key (S_{GK-MS}) and sends the *UT*, signature

and certificate ($Cert_{GK-MS}$) which was issued by the trusted *CA* to the *R's MS*.

- *R's MS* verifies the signature of the *GK's MS*, based on the certificate issued by the trusted *CA*. For *GK* selection of stage 1 & 2 the *UT* of the *GK* must be *legitimate or trusted*, else the verification will fail. In stage 3, the *UT* of the *GK* can also be *new* (relaxation of *UT new* is only given in stage 3, with email sending limit).

- *R's MS* picks the *UT* of the *R* (based on the *TR*), signs the *UT* using his private signature key (S_{R-MS}) and sends the *UT*, signature and his certificate ($Cert_{R-MS}$), issued by the trusted *CA* to *GK's MS*.

- *GK's MS* verifies the signature of the *R's MS*. The *UT* of the *R* can be *new, legitimate or trusted*. Only limitation on the *UT* is that the *R* must not be *illegitimate*.

- If the *GK* is selected for the first time, the *GK's MS* will generate *RSA* based key-pair (*PK, SK*) for the *GK* and send the *PK* to the *R's MS*. If the *GK* was already selected before for some other *R*, key-pair (*PK, SK*) will already exist for the *GK* and the *MS* will only send the *PK* to the *R's MS*.

- Once the *GK* has the *SK*, *GK's MS* will use the *SK* to issue a voucher ($(((User_{ID})_{hash})_{Sign-SK}$) to all the users of *GK's* community. These vouchers are added to the *VoucherList* of the users, along with the IDs of the *R* and the *GK* ($< R_{ID}, ... >, GK_{ID}$, and the $Voucher[((User_{ID})_{hash})_{Sign-SK}]$) and would be used later for communicating with the *R*. If a user is selected as GK for multiple *Rs*, its community users will not need multiple vouchers for each *R*. One voucher (from the *GK*) is enough to communicate with all the *Rs*. Only the *R's ID* is appended to the existing voucher entry in the *VoucherList* ($< R_{ID1}, R_{ID2}, ... >, GK_{ID}, Voucher[((User_{ID})_{hash})_{Sign-SK}]$).

- On receiving the *PK* of the *GK*, *R's MS* will add new entry *(PK, GKID)* to the *PKList* of the *R*. In the *GK* selection stage 3, if the *UT* of the *GK* is *new*, the *R's MS* will enforce a rate limit on the number of emails per day from the *GK* and his community user. Later, if the *UT* of the *GK* upgrades to *legitimate or trusted*, the *R' MS* will remove email sending limit after the verification of updated *UT*. This email limit is applied to restrict the number of spam messages sent by

a user before being identified as malicious. On the other hand, it will allow new honest users to send email outside their community.

In *LENS*, the use of *RSA* based key-pair *(PK, SK)* has many advantages over using symmetric keys between the *GKs* and their *Rs*. A *GK* requires a single key-pair, irrespective of the number of *Rs* it has. *LENS* evaluations § 7.2 reveal that a *R* on average requires only 10 to 31 *GKs* i.e. a *R* has to maintain only 10 to 31 *PKs* on average for his selected *GKs*. Moreover, the *GK* does not have to re-issue vouchers to his community members for every single *R*. A single voucher is sufficient for all the *R* for whom the *GK* is vouching. In *LENS*, the *PKs* are distributed directly by the *MS* of the *GK*. Hence, *LENS* does not require a public key infrastructure (*PKI*) for key management, distribution and verification and by doing so it avoids considerable overhead.

3.7.2 Recipient and *GK* on same *MS*

If the recipient and a *GK* are hosted by the same *MS*, the protocol works exactly the same without any need for *server authentication* (since all the messages are exchanged internally).

Chapter 4

LENS Prototype and Email Processing

Based on the conceptual system design introduced in chapter 3, this chapter presents the prototype implementation that integrates *LENS* with the existing email infrastructure. Furthermore, this chapter also explores the processing of emails with *LENS* and how spam can be prevented from transmission. Finally, this chapter concludes with the discussion on how *LENS* can be incrementally deployed.

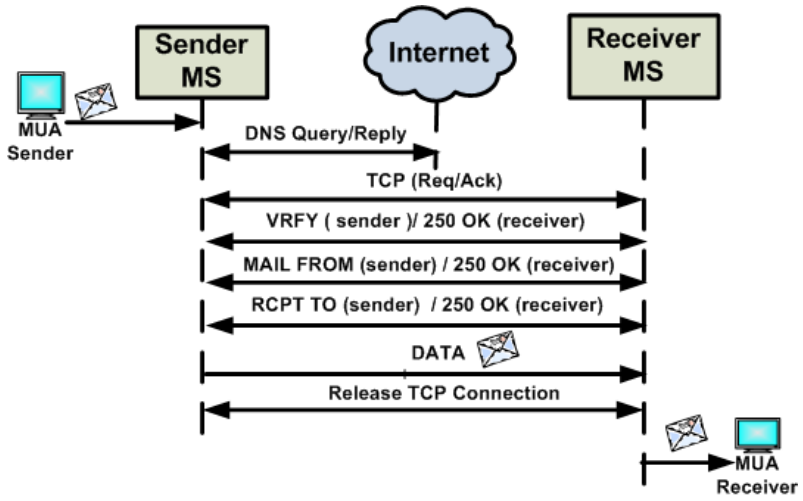

Figure 4.1: Standard email processing with SMTP.

4.1 Integration with Email Infrastructure

Figure 4.1 depicts the flow of an email i.e. from message creation, transport to delivery. Mail user agent (*MUA*), the email client of the sender, submits the email to its *MS* using *SMTP*. The sender's *MS* will look up the destination's mail exchanger record (*MX*) in the *DNS* server. The *DNS* server

finds the highest preference *MS* for the recipient and reports the name of the *MS* by returning a *MX* resource record. After this point, a *TCP* connection is established between the sender's and the recipient's MSs and the sender's *MS* sends the `VRFY:`, `MAIL FROM:` and `RCPT TO:` commands (one by one) to the recipient. With successful acknowledgment from the receiver side, the complete email (header and the body) is sent, and the *TCP* connection is released. The mail delivery agent (*MDA*) delivers the accepted email to a server for local mail delivery. Once delivered to the local *MS*, the mail is stored for batch retrieval by authenticated mail clients (*MUAs*) using *IMAP* or *POP*.

To use *LENS* for email processing during the *SMTP* transactions, additional steps are needed for its integration into the existing email infrastructure. This section covers the steps required to append vouchers on the email envelop (`RCPT TO:`) for the outbound email. It further presents how the vouchers can be extracted and verified by the receiving *MTA*. Without any modification to the existing *SMTP* specifications, *LENS* based email filter (using *CommList* and vouchers issued by the *GK*) runs as an independent daemon (like spamd for *SpamAssassin*: spamassassin.apache.org) to monitor the *SMTP* transaction and based on the results takes different actions.

This implementation used the Mutt mail client, the MailAvenger SMTP daemon [5], and Postfix [7] as an *MTA*. Postfix is a widely used open source *MTA* that is available for all major UNIX-based operating systems. For the processing of inbound emails MailAvenger has been used [5]. MailAvenger is a *MTA* independent *SMTP* daemon that allows application of filters on an email during the *SMTP* transaction, which meets our requirements. Figure 4.2 gives an overview of the processing of inbound and outbound emails.

4.1.1 Integration on the sending side

LENS utilizes Postfix as an *MTA* for the processing of outbound emails. In Postfix, *SMTP* transactions are implemented in the `smtp_loop()` function within the `smtp_proto.c` file. The following code listing outlines the functions definition of `smtp_loop()`.

```
static int smtp_loop(SMTP_STATE *state, NOCLOBBER int
                     send_state, NOCLOBBER int recv_state)
```

With respect to *LENS*, the relevant part of the `smtp_loop()` function is the realization of `RCPT TO:`. The following code excerpt outlines the original

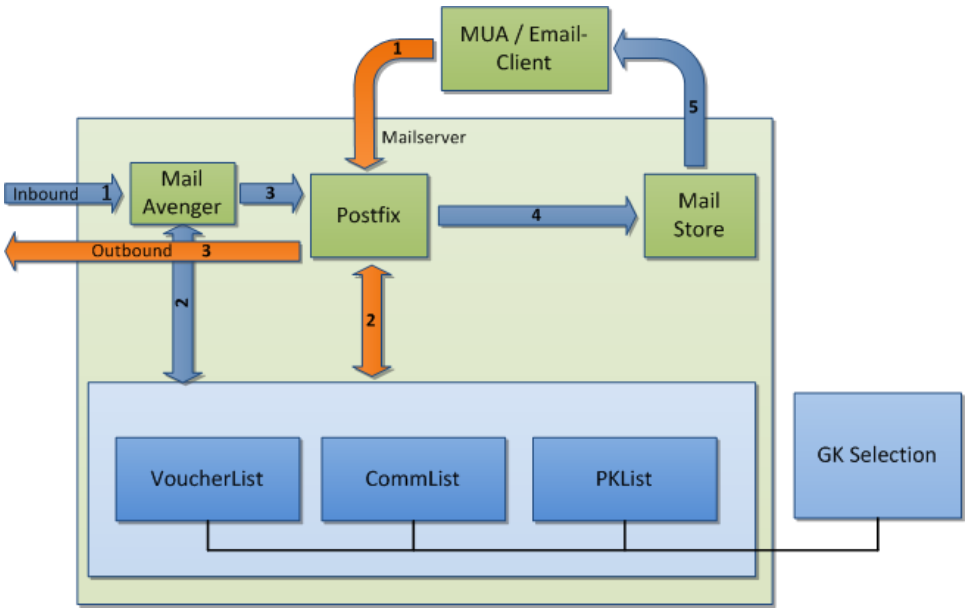

Figure 4.2: An overview of the processing of inbound and outbound emails with *LENS*.

code fragment that handle the `RCPT TO:` prior to the integration of *LENS*.

```
case SMTP_STATE_RCPT:
  rcpt = request->rcpt_list.info + send_rcpt;
  REWRITE_ADDRESS(session->scratch2, rcpt->address);
  QUOTE_ADDRESS(session->scratch, vstring_str(session->scratch2));
  vstring_sprintf(next_command, "RCPT TO:<%s>",
                  vstring_str(session->scratch));
```

For the integration of *LENS* with Postfix on the sending side, the processing of `RCPT TO:` command has to be customized in two aspects. In the first step, the source code is modified to check the *CommList* of the sender if it has an entry for the recipient. If the recipient is in the sender's community the `RCPT TO:` command will be executed without any modification as follows (couple of lines are skipped in the listing for simplicity).

```
case SMTP_STATE_RCPT:
  rcpt = request->rcpt_list.info + send_rcpt;
  REWRITE_ADDRESS(session->scratch2, rcpt->address);
  QUOTE_ADDRESS(session->scratch, vstring_str(session->scratch2));
```

```
  //read the CommList

  f=fopen(commlist_path,"r");
    while ( fgets( buffer, 512, f) != NULL ) /* read a line */
     {
       word = strtok(buffer, " ?");
       if(strcmp(word,vstring_str(session->scratch)) == 0)
        {
         // if recipient is in the CommList
         InCommList = 'T';
          break;
        }
      }
   fclose (f);

   if (InCommList == 'T')
    {
      //execute the RCPT TO without any modifications
      vstring_sprintf(next_command, "RCPT TO:<%s>",
                      vstring_str(session->scratch));
    }
```

If the community lookup fails, the second step is to look for a valid voucher issued, to the sender, by the trusted *GK's MS* to communicate with the recipient. Upon the successful retrieval, the resulting voucher is appended to `RCPT TO:` as an additional parameter.

```
if (InCommList == 'T')
    vstring_sprintf(next_command, "RCPT TO:<%s>",
                    vstring_str(session->scratch));
else
  {
    // look for the Voucher
    f=fopen(Voucherlist_path,"r");
    if(strcmp(word,vstring_str(session->scratch)) == 0)
    {
      InVoucherList = 'T';
      // Voucher exist, pick the GKID and Voucher
```

```
      break;
     }
     fclose (f);
    }


if (InVoucherList == 'T')
   {
      #Append GKID and Voucher as Additional Parameters to RCPT TO
      vstring_sprintf(next_command, "RCPT TO:<%s> voucher=%v gk=%s",
                 vstring_str(session->scratch),voucher,gk);
   }
```

The additional arguments to `vstring_sprintf()` function reflects the appending of additional parameter to `RCPT TO:` command. At the end of step two, if there is no valid voucher for the recipient, the *GK* selection module will be invoked to execute GK selection in stage 3 using `system()` call. The `system()` function is part of the C standard library and can be used to execute an external program. Here it is used to minimize changes to the Postfix source-code for this prototype. On successful completion of GK selection in stage 3, the withheld `RCPT TO:` is sent out after appending it with the newly acquired voucher. The flowchart in figure 4.3 represents the processing of outbound `RCPT TO:` command.

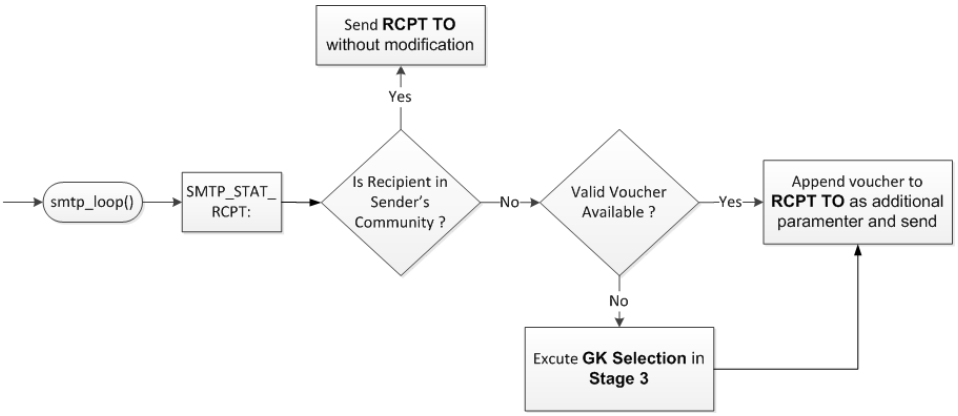

Figure 4.3: The processing of outbound `RCPT TO:` command.

In order to maintain compatibility with non-*LENS* servers at the receiving side, the execution of GK selection (stage 3) should be skipped after the

failure of voucher lookup and a normal `RCPT TO:` command should be used.

4.1.2 Integration on the receiving side

As stated above, *LENS* use MailAvenger for the processing of inbound emails. The main advantage of MailAvenger is that, by default, it supports direct access to the SMTP transactions. This allows application of filters on an email during the SMTP transaction, as per our requirements. In MailAvenger, the core working of SMTP is handled in the `smtpd.c` file and the `RCPT TO:` functionality is implemented in the `cmd_rcpt()` function.

```
void smtpd::cmd_rcpt (str cmd, str arg)
```

The processing of inbound `RCPT TO:` command can be broken down in two steps. In the first step, a community based filter is applied to check if the sender is in the recipient's community. This is implemented in the `CommFilter()` function. The function takes in the sender's and recipient's addresses as an argument and returns a string "yes" or "no".

```
str CommFilter (str recipient, str sender)
{
  char commlist_path[512] = "";
  char buffer[512];
  str word;
  strcat(commlist_path, recipient);
  FILE *f;
  f= fopen(commlist_path,"r");
  while ( fgets( buffer, 512, f) != NULL )
  {
word = strtok(buffer, " ?\n");
if(word == sender)
  return "yes";
  }
  fclose (f);
  return "no";
}
```

The result of this community lookup is then exported as an environment variable `COMMUNITY_FILTER` in the same file (`smtpd.c`) as follows.

```
envp->push_back (strbuf ("COMMUNITY_FILTER=") << communityfilter);
```

MailAvenger allows users and administrators of a *MS* to use shell scripts for defining their filtering rules. To access properties of inbound mails in these scripts, for example the sender's address or the results of an SPF check, MailAvenger exports various environment variables. The same approach is used for *LENS*, more precisely the result of the community filter and the voucher verification are exported to a shell script using environment variables.

If the community lookup fails, the next step is to extract the voucher of an inbound email from `RCPT TO:` and verify it. The `extract_substr()` function is used to extract the voucher appended as additional parameter to the inbound `RCPT TO:` command.

```
str addr = extract_addr (arg, "to:");
str voucher = extract_substr (arg, 'V');
str gatekeeper = extract_substr (arg, 'G');
```

The function `extract_substr()` has been defined in `addrparse.C` and can be used to find a substring of the string given as the first parameter. The parameter `V` and `G` defines that the voucher and GKID should be extracted, so `extract_substr()` will search for the substring `voucher=` and `gk=` in `arg` and return the values. These values are then passed on to the `verify_voucher()` function for the verification of the voucher.

```
if (voucher != "NULL")
{
 IsVoucher = "yes"; //voucher exist
 verify = verify_voucher(voucher, fromaddr, gatekeeper, addr);
 if (verify == 1)
   VoucherVrfy = "1";

 else if (verify == -1)
   VoucherVrfy = "-1";

 else if (verify == 0)
   VoucherVrfy = "0";

 else if (verify == 2)
   VoucherVrfy = "2";
}
```

For the verification of vouchers, the `verify_voucher()` function use OpenSSL API to access standard implementation of public-key cryptosystem (RSA) and hash (SHA-1) functions. The result of voucher verification is then exported in two environment variables as follows.

```
envp->push_back (strbuf ("IS_VOUCHER=") << IsVoucher);
envp->push_back (strbuf ("IS_VOUCHER_VERIFIED=") << VoucherVrfy);
```

At the end of `RCPT TO:` processing, MailAvenger automatically execute the script placed in the `/etc/avenger/default` file. Using the environment variables, this script defines filtering rules to determine whether the emails should be accepted or rejected. Alternatively the script can be placed in the home directory of any user, for example in `/home/shameed/.avenger/rcpt` for the user shameed. With this setup, the script is only executed for emails which are addressed to the owner of that directory, whereas the first setup executes it for all incoming emails. Following is the example of the `default` shell script used for *LENS*.

```
if [ $IS_VOUCHER = 'yes' ]; then
    if [ $IS_VOUCHER_VERIFIED = '1' ]; then
        accept "$SENDER has valid GK Voucher"

    elif [ $IS_VOUCHER_VERIFIED = '2' ]; then
        reject "$SENDER has UnAuthorized GK"
    else
reject "$SENDER has invalid GK Voucher"
    fi

elif [ $COMMUNITY_FILTER = 'yes' ]; then
   accept "$SENDER is in the CommList of $RECIPIENT"

else
   reject "$SENDER is not in the CommList of $RECIPIENT
           and doesnt have valid voucher"
fi
```

The flowchart in figure 4.4 represents the processing of inbound `RCPT TO:` command.

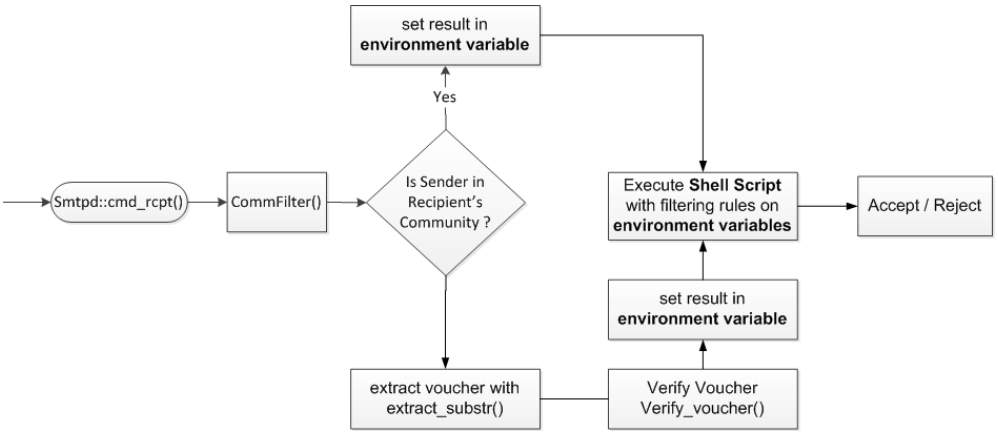

Figure 4.4: The processing of inbound `RCPT TO:` command.

4.1.3 Conclusion

The design of *LENS* allows an easy integration for the new adopters. For inbound mails, MailAvenger can be used to formulate rules that can efficiently access the community-based and voucher-based filtering. Furthermore, the required changes to the source-code of Postfix and Mail Avenger are minimal.

4.2 Email Processing With *LENS*

With *LENS*, email processing (cf. figure 4.5) can fall into one of three categories:

1. When a message is sent to any *recipient* **within the community**, the *recipient's MS* will verify the *sender* against the *recipient's CommList* and place the message in the mailbox.

2. If a message is sent to a *recipient* **outside the *sender's* community**, the *sender's MS* will bind a voucher, from authorized *GK* along with the message. On reception, the *MS* will verify the voucher using the *PK* stored in *PKList* against the *GKID* and place the message in the *recipient's* mailbox.

3. If a new message is intended for a *recipient* **outside the *sender's* community and with no voucher issued by any *GK***, the *sender's MS* will hold the message and start a *GK* selection procedure (stage

3). On successful completion, the *sender's MS* will now bind a voucher with the withheld message and send it out. At reception, the *MS* will verify the voucher using the *PK* stored in *PKList* against the *GKID* and place the message in the *recipient's* mailbox.

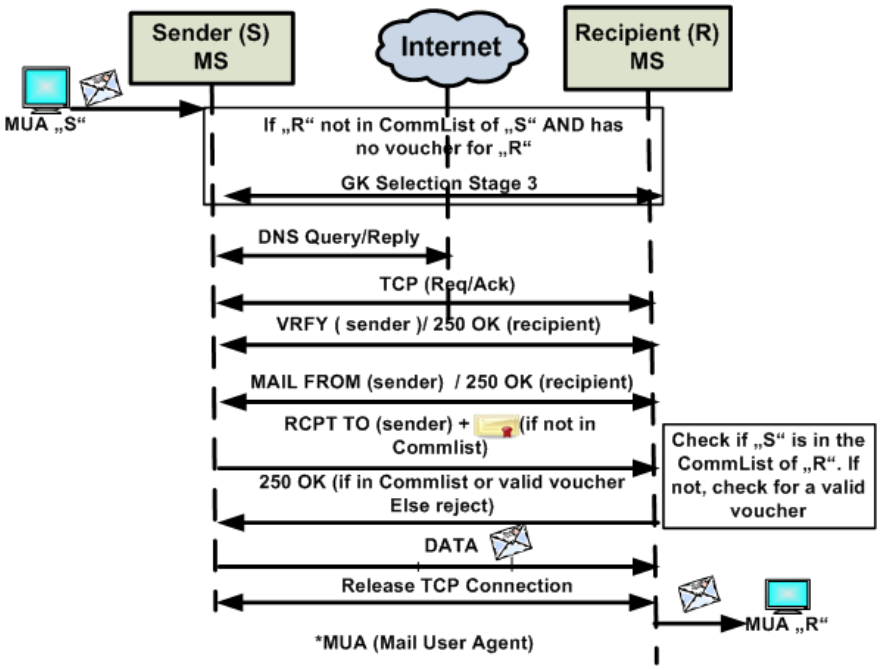

Figure 4.5: Email processing with *LENS*.

4.3 Prevention of Spam Transmission

One of the main contributions of *LENS* is that it prevents the transmission of spam across the network. Let us consider that the sender's and recipient's *MS* have already established a *TCP* connection. Now, when the sender's *MS* will send the `RCPT TO:` command, it will also append the voucher and issuing *GK's* ID (for e.g. *RCPT TO:* <example@abc.com>Voucher = 1f2a91aa236d0012 GK=gk@example.com) as additional *rcpt-parameters*, if the recipient is not in the sender's community (figure 4.5). At the recipient's end, the *MS* verifies if the sender is a community member or has a valid voucher from an authorized *GK*. Failure of the verification results in

the termination of the *TCP* connection by the *recipient's MS* and the transmission of email (header and body) will not take place, thereby preventing the unwanted message from being transmitted.

According to the current draft standard of *SMTP* [49] using additional rcpt-parameters is optional and contemporary *SMTP* implementations ***must*** support it as basic extension mechanisms. The *SMTP* server not obliged to understand the additional rcpt-parameters simply ignores them.

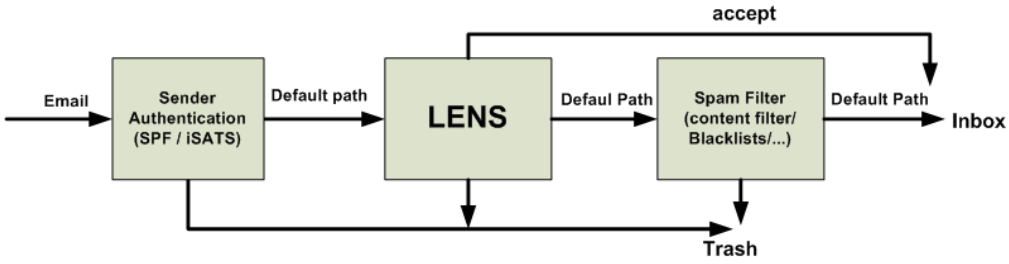

Figure 4.6: Complementary *LENS* and existing spam filters

4.4 Incremental Deployment

It is easy to integrate *LENS* into current SMTP servers. Inevitably, when deployed, some users will adopt *LENS* before others. Until every user is familiar with *LENS*, it will run complementarily with the existing spam filters. See figure 5.4, after sender authentication *LENS* is first to examine inbound email. Incremental deployment will have four communication scenarios as follows.

1. ***Sender* and *Recipient* both have *LENS*:** email processing will follow the procedure discussed in § 4.2 (also see figure 4.5)
2. **Only *Sender* has *LENS*:** *Sender* will send an email as usual and the *MS* on the *recipient* side will process the email with its existing spam filters.
3. **Only *Recipient* has *LENS*:** after performing a check for existence of community or voucher, *LENS* will pass the email to the existing spam filters.
4. **Both *Sender* and *Recipient* do not have *LENS*:** emails will be process according to the existing mechanisms deployed at the *sender's* and *recipient's MS*.

Based on our evaluations around 81%-89% (§ 7.2) of the emails are from the community members. With the increase in adaptation/deployment of *LENS*, the user's community will grow and their reachability via the *GKs* will increase. With global deployment *LENS* is envisioned to stop the transmission of spam at the first place. Even with partial deployment, the *senders* and *recipients* deploying *LENS* will benefit in two ways. First, all the emails from the user's community and with valid *GK* vouchers will be processed by *LENS*, which is significantly efficient than SpamAssassin (see § 7.4). Second, less number of emails will pass through existing content-based filters, and avoid the chances of false positives and false negatives due to imprecise content signatures. These benefits will also help motivating the deployment of *LENS*.

Chapter 5

Complementary Email Sender Authentication

SMTP does not authenticate sender address by default. Therefore, spammers can easily launch a spam attack with forged ***from:*** addresses as if they are from the recipient's community. To address this problem a new crypto-based email sender authentication mechanism known as *iSATS* has been developed as a minor contribution to this thesis. *iSATS* is complementary to *LENS* and provides a reliable way to bind the identity of a legitimate sender to an email. The mechanism has made it reasonably hard for a spammer to adopt or bypass the system without getting noticed.

LENS can also utilize *SPF* [75], for sender authentication. *SPF* is the most adopted email sender authentication protocol effectively being used in existing email systems. However, the advantage of *iSATS* over *SPF* is that it forms a close system that strongly binds the sender's identity. This makes it hard for spammers to adopt the system without becoming visible. *SPF* on the other hand can be easily abused and according to [60] 20% of the spamming domains have already adopted *SPF*. Nevertheless, *iSATS* is proposed as a complementary service to *LENS* and it would be a design choice for email service providers (or domain administration) to go with *iSATS* or use other existing sender authentication schemes. The following sections of this chapter discuss the *iSATS* protocol design, prototype and evaluations.

5.1 *iSATS* Design

iSATS is a crypto-based email sender authentication system that operates on the *SMTP* envelop, in particular on `MAIL FROM:` command, to perform domain level authentication during the SMTP transaction. *iSATS* leverages identity based signature (*IBS*) using identity-based public key cryptography (*IBC*) [20, 27] to authenticate the identity of an email sender. Compared to traditional public key cryptography, *IBC* eases the burden of managing and

distributing public keys, since publicly available unique identities are used as public keys.

iSATS requires a trusted authority (*TA*) also known as private key generator (*PKG*), responsible for issuing secret key (*SK*) and system parameters. The *TA* is also responsible to thoroughly verify the identity of a domain before issuing *SK*. This verification strongly binds the identity of the domain owner to its domain and makes it hard for the spammer to adopt *iSATS*, unless they are willing to give away their identity.

iSATS requires the sender's *MTA* to generate a signature using SK of the domain and appends it with *SMTP* envelop (`MAIL FROM:`). This enables the recipient's *MTA* to quickly authenticate the sender by verifying the appended signature. Any invalid connection is terminated right away, saving valuable resources both at the *MTA* and in the network. Unlike *SPF* and *DKIM*, email forwarding and munging of message along the transit is not a problem in *iSATS*.

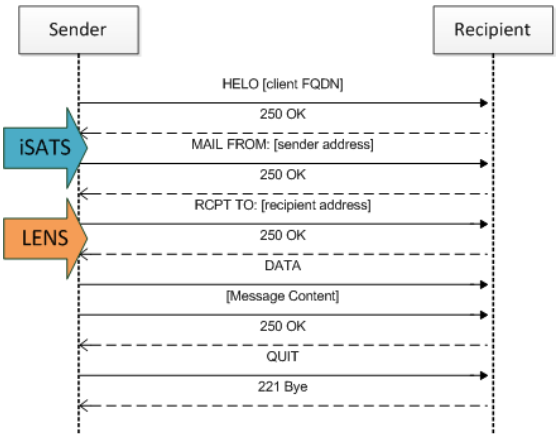

Figure 5.1: The SMTP transactions where *iSATS* and *LENS* operates independently.

5.1.1 Basic requirements

From the outset, we sought a solution that works well with the current, entrenched email system. This means that our proposed system should:

- Work as an optional addition to standard mail clients or servers, and continue to support popular means of accessing mail (e.g. IMAP / POP / Webmail).

- Be incrementally deployable.
- Remains transparent to end users.

The functionality of *iSATS* can be divided into four steps: 1) setup, 2) identity verification and secret key extraction, 3) signature generation and 4) signature verification. Each of these steps are discussed in the following sections.

5.1.2 Setup

This step is executed only once in the beginning and marks the creation of a whole *IBC* environment by a *TA*. The setup results in generation of a *Master Key* and *System Parameters*. *Master key* is kept secret by the *TA* and it is used to generate *SK* for the domains based on their identity. *System parameters* on the other hand are publicly available.

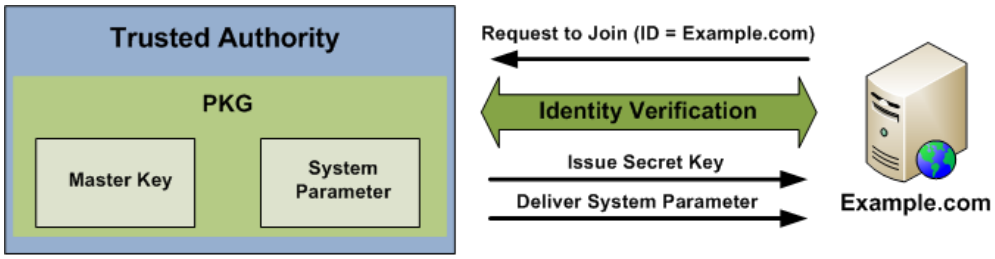

Figure 5.2: The process of domain joining the *iSATS*.

5.1.3 Identity verification and secret key extraction

This step is initiated when any domain wants to become part of *iSATS* and requests a *SK* (see figure 5.2). *iSATS* represents a closed system, in a sense that domains are not automatically added to the system. In order for the domains to get included into *iSATS* the domains would have to be verified by the *TA*. *iSATS* is envisioned to provide extended validation of the domain's identity. Identity verification with *Extended Validation* [35] provides high-level of security to clearly identify a domain's organizational identity. This will help bind the owner's identity to the identity of the domain and will make misbehaving domains visible. Most well-known webmail providers and websites tend to have SSL certificates with extended validation, so this requirement is not excessive.

After identity verification, the *TA* will issue system parameters and a *SK* corresponding to the domain name which is also its identity and *PK*.

This meets the requirements of the system, as the domain name is a unique identity for the domain and is publicly available to all parties.

5.1.4 Signature Generation

This step is executed when a user wants to send an email. The *MTA* will generate a signature for the sending user's email address (e.g. alice@example.com) using the *SK* and system parameters of the domain. This signature is then appended to `MAIL FROM:` command as an additional parameter (see figure 5.3). The use of additional parameters in `MAIL FROM:` is allowed and in line with the current SMTP specifications [49].

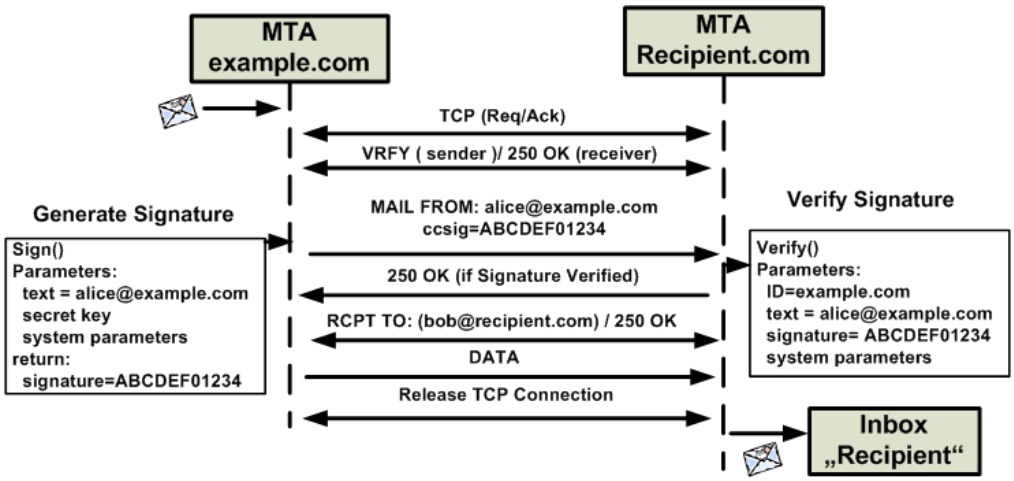

Figure 5.3: Email processing with *iSATS*.

5.1.5 Signature Verification

On receiving the `MAIL FROM:` command, the *MTA* on the receiving side will verify the signature. For verification the *MTA* will use the public system parameters, signature (extracted from `MAIL FROM:`), signed text i.e. sending user's email address from `MAIL FROM:` and the domain name of the sender as the public key (see figure 5.3). The entire verification process is completed before replying to `MAIL FROM:`, which gives recipients the option to reject the message before its content is sent.

5.2 Discussion

5.2.1 *iSATS* complementary to *LENS*

Both *iSATS* and *LENS* operate on the email envelope i.e. on `MAIL FROM:` and `RCPT TO:` transactions of the *SMTP* protocol. This helps in filtering the emails at an early stage before the content is transferred. *iSATS* is proposed as a complementary service to *LENS* and it will be the decision of the email service provider (or domain administration) to use *iSATS* or use an existing sender authentication scheme. If both the systems are used together, the sever authentication can be skipped during the legitimacy verification of *GK* (§ 3.7) in *LENS*, since the identity of the *MSs* is already validated by *PKG/TA* at the time of joining *iSATS* (§ 5.1.3).

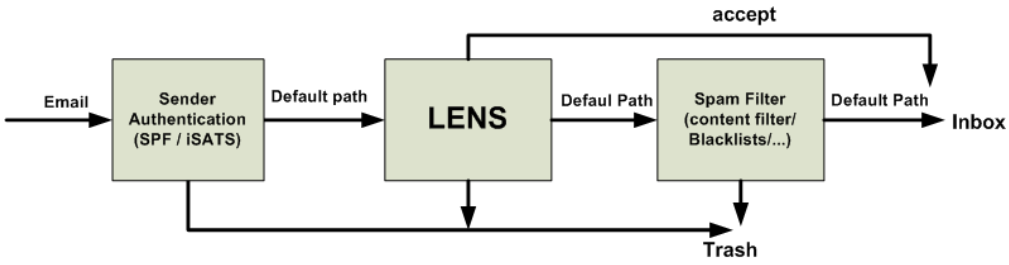

Figure 5.4: *iSATS* complementary to *LENS* and existing spam filters.

5.2.2 Email forwarding

In email forwarding, it is a common practice not to change the return path or `MAIL FROM:` message envelop. This is an Achilles' heel in IP-based *SPF* [76]. However, for *iSATS*, message forwarding is not a problem as long as `MAIL FROM:` command remains intact.

5.2.3 Message munging

iSATS does not operate on the content of an email. If the content of the message is altered during transit by a mailing list (which is a common practice) it will have no effect on *iSATS*, unlike *DKIM* [76].

5.2.4 Security of TA

In *iSATS*, the security of the *TA* is very crucial and if an attacker is able to obtain the *TA's* master key, he would be able to issue *SKs* and generate

valid signatures. It is synonymous to securing a *certification authority* (such as VeriSign) for the legitimacy of the issued certificates. In order to secure central *TA*, Boneh and Franklin [20] introduce a concept for distributing the *TA* (*PKG*) in such a manner that the master key is distributed over a set of nodes so that each node has no information over the key itself. Domains can extract their *SK* by obtaining partial keys from a subset of these nodes, where the subset must be bigger than a certain threshold. This kind of distribution minimizes the effect of DDoS attacks. Further discussion of the security of the *TA* is beyond the scope of this thesis and will be considered in future work.

5.2.5 Attack on secret key of domain

Protection of *SK* is the responsibility of the *MTA* or the Domain. Attacks related to key thefts are synonymous to hacking the domain and the corresponding defense mechanisms are beyond the scope of this thesis. Hence, the discussion on key theft and revocation mechanisms has been left as a future work.

5.2.6 Signature re-use or Mis-use

In order to avoid reuse or misuse of signatures by a potential attacker it is recommended to use a unique signature for each message. This can be done by using a nonce (a unique number used only once) and instead of just signing the sender's email id the *MTA* can sign email ID + nonce. The nonce can be composed of a time stamp value, message ID or a combination of both. With this addition, the new `MAIL FROM:` command will be *MAIL FROM*: <alice@example.com>, ccsig = Sign (alice@example.com + nonce), nonce.

5.2.7 Sender reputation

Email sender authentication systems only authenticate the identity of the email senders at the domain level. For the legitimacy of the domain it is recommended that each domain maintain a local reputation for the domains that are sending emails. As part of future work, sender reputation can also be centralized at the *TA* level, based on the feedback of individual legitimate domains.

Nowadays, it is also a common practice in legitimate domains, ISPs and major web-mail providers to run bot detectors against non-human automatic

account creation and impose an email sending limit between 100 to 1000 recipients/day [2].

5.3 *iSATS* PROTOTYPE and EVALUATION

A basic prototype implementation of *iSATS* has been developed, which includes a server implementation that offers the basic functionality of a *TA* like system setup (generation of system parameters and master key in the beginning), extraction and distribution of *SK* to the requesting domain.

iSATS based email processing (ID based signature generation and verification) has been integrated during the *SMTP* transactions with the Mutt mail client, the MailAvenger *SMTP* daemon (mailavenger.org), and the Postfix *MTA* (postfix.org). The prototype is based on Cha-Cheon *IBS* scheme [24] and utilize Pairing-Based Cryptographic [54] library to handle the cryptographic functions. Instead of modifying the *SMTP* implementation *iSATS* runs as an independent daemon (like spamd for SpamAssassin: spamassassin.apache.org) to monitor the *SMTP* transaction and takes different actions based on the results of this monitoring.

The potential bottlenecks of *iSATS* are the computationally expensive tasks, specifically signature generation and verification, and the extraction of *SKs* by the *TA*. To evaluate the influence of *iSATS* on email processing, the performance of the tasks mentioned above are evaluated on different platforms with varying computational capabilities. To put results in perspective, the email processing performance without *iSATS* is analyzed on the same systems and provided as a reference. The system specification for the evaluations are as follows:

Workstation: A 2.0 GHz Intel Core 2 Duo machine with 3 GB RAM.

Netbook: A single core 1.6 GHz Intel Atom CPU with 1 GB RAM. The netbook is used to evaluate *iSATS* in resource constraint environment.

Virtual Machine (VM): For the initial implementation and testing a virtual machine environment has been used, consisting of two virtualized Ubuntu Server installations on a VMware ESXihost. Each VM has been assigned a 2.4 GHz quad-core Intel Xeon CPU and 256 MB RAM. To simulate a system with low performance the CPU clock has been limited to 800 MHz , where denoted.

During the evaluations the physical systems are connected over a *LAN*, whereas the *VMs* are connected over a virtual network.

5.3.1 Performance of *TA* in *SK* extraction

For the performance of *TA*, the ability of a *TA* to handle the *SK* extraction requests has been studied and throughput and processing delay are used as a performance metrics to evaluate *SK* extraction.

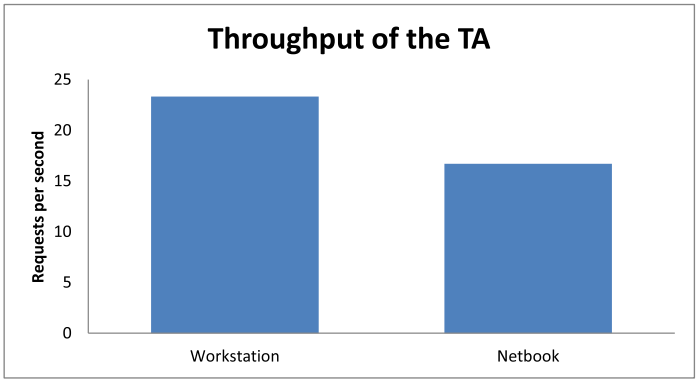

Figure 5.5: The throughput of secret key generation.

Throughput

In order to measure the throughput of *TA's SK* extraction, the servers at the netbook and the workstation are bombarded with 1000 *SK* requests as rapidly as possible. Even with bombardment of *SK* requests both the netbook and workstation were able to successfully handle 17 and 23 requests per second respectively (see figure 5.5). It is also interesting to notice that both the servers eventually finished the *SK* extraction for all the requests received.

Processing delay

On average it takes around 43 ms to receive a *SK* with workstation as a *TA*. For netbook the average processing delay for *SK* request is around 68 ms (see figure 5.6). Since the experiments were done over *LAN*, it does not involve any latency due to geographic distance between domain requesting the *SK* and the *TA*. Ideally each domain is expected to query the *TA* just once for a *SK*, therefore the processing delay introduced due to the *SK* request will become insignificant overtime.

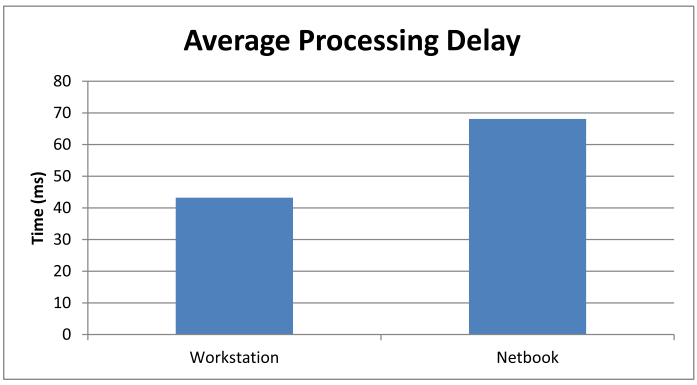

Figure 5.6: The average processing delay of SK generation.

5.3.2 Performance of email processing with *iSATS*

For evaluating the system performance of email processing with *iSATS*, standard email processing system has been augmented with *iSATS* implementation (ID based signature generation and verification), and deployed on the netbook and workstation over the *LAN*. Different experiments were performed to study the impact on throughput, delay, and CPU and memory consumption. *VM* setup of *iSATS* is also used in the experiments.

Throughput

To evaluate the throughput, three different experiments were run on different setups (workstation, netbook, 2.4 GHz *VM* and 800 MHz VM) both with and without running *iSATS* during the experiments. For the first experiment, a burst of 1000 messages are sent, without any delay in between, to the respective systems. For the second and third experiment a delay of 0.1 and 0.2 seconds has been introduced between each message.

On the workstation, the throughput with and without *iSATS* remained identical for all three experiments (See figure 5.7). In the first experiment with no delays, the throughput on the netbook declines significantly from 15.28 to 5.67 messages per second after *iSATS* was activated. However, with the addition of delays in experiment 2 and 3 the throughput remains identical at 5+ messages per second with and without *iSATS*. On the VMs the influence of *iSATS* is negligible and the throughput is almost identical for experiment 2 and 3 on different *CPU* resources. The disproportional performance of the virtual machines is due to the absence of a physical *LAN* connection. Overall, the results indicate that *iSATS* does not affect the

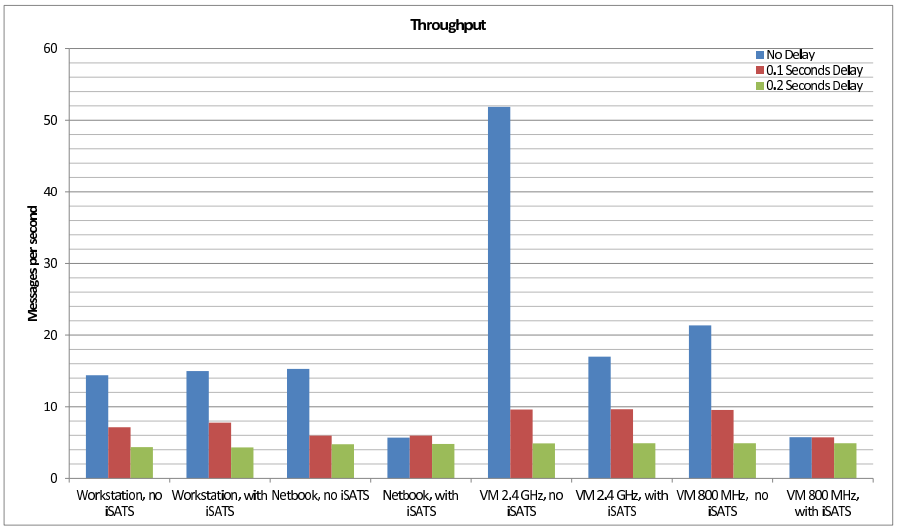

Figure 5.7: The average email throughput on different setups.

throughput significantly if the computing capabilities are sufficient.

Processing delay

In order to evaluate the processing delay, 1000 messages are sent with an interval of 0.5s on the servers running *iSATS* with different computational setups. The results are averaged in figure 5.8 and the values show the extra time that was needed to process an email using *iSATS*. As expected, the processing delay on the netbook is highest at an average of 185 ms. The delay on the workstation is only 62 ms on average, or about one third of the netbook. Nevertheless, the processing cost of an email with *iSATS* is minimal.

CPU and Memory usage

The effect of *iSATS* on *CPU* and memory usage has been analyzed over a time span of 10 minutes, while receiving a message every 0.3 s. This experiment is also performed without *iSATS* to provide a reference point. The *CPU* usage is almost identical on all systems, irrespective of *iSATS* (see figure 5.9). The spikes on the netbook are assumed to be caused by system processes that have nothing to do with email processing.

The memory usage is also minimal and it is hard to notice any difference when *iSATS* is running (see figure 5.10). On the netbook, the higher

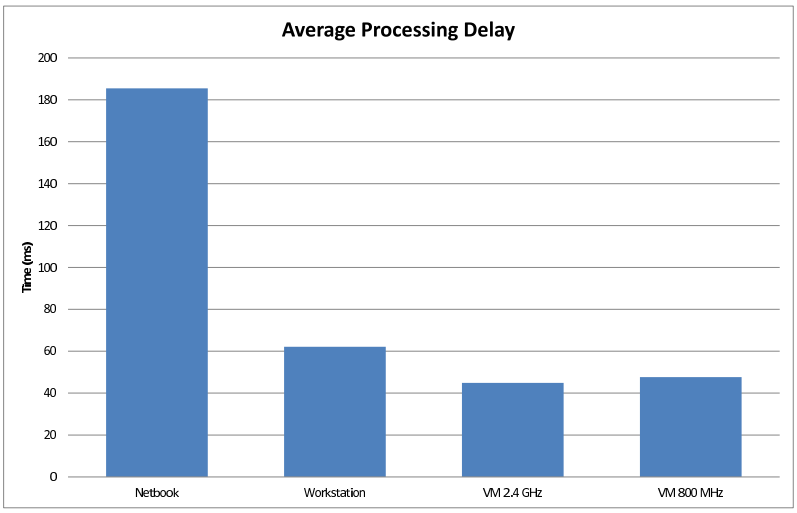

Figure 5.8: The average email processing delay with *iSATS*.

memory usage as compared to workstation is primarily due to its limited resources. In the end it can be concluded that *iSATS* has no significant effect on *CPU* and memory resources.

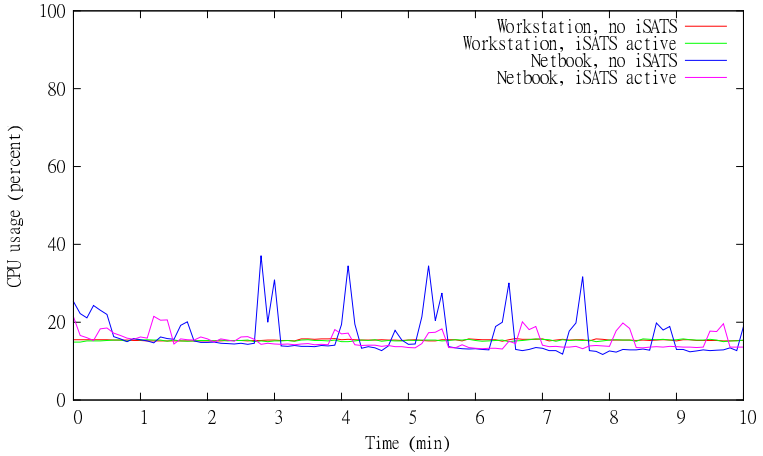

Figure 5.9: The cpu usage during email processing.

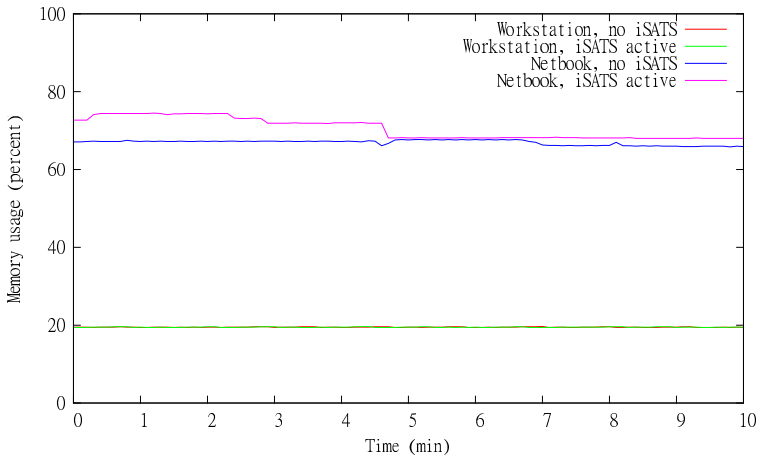

Figure 5.10: The memory usage during email processing.

Based on the results presented in this section, it has been observed that *iSATS* is computationally efficient. *iSATS* is fast in processing emails and even on a low end processing machines the overheads are in milliseconds. The *TA* also scales efficiently with *SK* request bursts and has a overhead within milliseconds even on a low end processing machine.

5.4 Conclusion

This chapter introduced *iSATS*, a new email sender authentication system that leverages identity based signatures for stronger sender authenticity than existing solutions. With the help of a trusted authority, *iSATS* forms a closed system that provides a reliable and easy way to bind the identity of a legitimate sender to an email. On the other hand, it is hard for the spammer to adopt the system without getting noticed. In addition to this, *iSATS* operates on email envelope, specifically on `MAIL FROM:` command, which makes it easy to reject spam before receiving the actual content.

iSATS is complementary to *LENS* and the evaluation of the prototype shows the feasibility of the proposed system, and also reveals how it can be integrated with tools commonly used throughout the email infrastructure. The results show promising performance with low processing overhead on different platforms.

Chapter 6

Security Concerns

6.1 False Positives and Negatives

In *LENS* spam prevention is based on social communities and *GK* formation. Since *LENS* does not rely on content filtering, it does not encounter any false positive or negatives generated by imprecise content signatures. However, there is a possibility that due to false deduction of social context, malicious users can become part of the community and hence result in false-negatives (i.e. spam messages). But even with the presence of malicious users, *LENS* would not have false positives (i.e. non-delivery of good emails).

6.2 Compromised User

If a user (who may also be a *GK*) is compromised (say someone steals his identity), it will only have a local effect within the community. The effect is temporary and only lasts until the victimized user broadcasts the incident using his other Ids (maybe through friends or word of mouth) or claims back his ownership from the email service provider. Let us suppose that the victimized user is unable to reclaim the ownership of his Id. In that case, the user can always request its community to abandon his compromised Id and the *MS* will remove all the data associated with the compromised Id from the *PKList* and *VoucherList*. Hence, the attacker would not be able to harm the system on a large scale as the *SKs* and vouchers are handled internally by the *MS*.

6.3 Malicious User

If spam originates from a user associated with *GK* (one of the community member of the *GK* is malicious), the recipient can report the offending user as a spammer. Upon receiving certain threshold of reports, the *MS* of the recipient will either add the offending user to the revocation list (remote user) or mark the user (local user) as *illegitimate* to prevent further

spamming. One user's bad behavior will not prevent all other users associated with the *GK* from communicating with the recipient. On a different note, let us suppose that a *GK* becomes malicious (or violates the protocol) and start vouching for illegitimate (spammers) users within and beyond its community. This will result in transmission of spam messages towards the recipient, due to which the recipient will have to report multiple offending users as spammers to the *MS*. If the *MS* receives several spam reports for the users associated with the *GK*, the recipient's *MS* will revoke the *GK* by deleting corresponding entry in the *PKList*, resulting in invalidating all the vouching from the *GK*.

6.4 Trust Farming

A Spammer cannot increase his own *TR* by giving votes to others i.e. by sending emails. However, spammers can launch a trust farming attack on *LENS*, where a spammer votes for another spammer(s) to increase their rank. This attack is comparable to link farming on the Internet to attack the PageRank. However, extensive work has already been done to identify and neutralize such attacks on social reputation schemes. Mature solutions like [77] can be used orthogonally in *LENS* to identify and protect against trust farming.

6.5 Human Spamming using *GK* Selection in Stage 3

Stage 3 of *GK* Selection protocol allows new users to send emails to complete strangers. A new user who is also a spammer can exploit stage 3 to become a *GK* and spam the recipient. However, due to rate limit on emails from *GKs*, whose *UT* is *new* (*NU*), the scale of the attack will be small. Furthermore, creation of arbitrary new IDs (if a spammer care about being reported and does not reuse old ID ever) and repetition of stage 3 will require substantial human involvement. This will incur costs that are against the normal spamming model.

6.6 Voucher Misuse and Revocation

The vouchers issued by the *GK* are bound with the identities (userid+domain for e.g. abc@example.com) of the community members and can only be used

by the user to whom they are issued. Current version of *LENS* does not impose any expiration date on the vouchers. Vouchers become invalid or get revoked/terminated by the *MS* based on four conditions: 1) when a *GK* removes a user from his community, the *MS* will also remove the associated vouchers (even if the user gets hold of the voucher, he cannot use the voucher on his own after being removed), 2) the user himself gets out of *GK's* community, 3) *UT* of the user becomes negative, and 4) if *recipient* revokes the *GK*, all the associated vouchers will become invalid.

6.7 Malicious GK Faking UserType (*UT*)

By design GK has no control in manipulating the *MS* to send false *UT*. There exist two possibilities where an illegitimate *GK* can send a false UT to the recipient. In the first case, a malicious *GK* can host a private *MS*, certified by a *trusted certification authority* and configure it to always send *legitimate* for his *UT*. However, by doing so the malicious *GK* will also become visible. This is not an option for spammers as it makes them vulnerable to legal actions. In the second scenario, the *MS* could be compromised by a malicious *GK*. This is synonymous to hacking the *MS* and the corresponding defense mechanisms are beyond the scope of this thesis.

6.8 Key Theft

Protection of keys is the responsibility of the *MSs*. Attacks related to key thefts is synonymous to hacking the *MS* and the corresponding defense mechanisms are beyond the scope of this thesis.

6.9 *LENS:* A Self-Correcting System

It is true that if a spammer can get a *GK* to vouch for him, he can spam all the recipients who authorize that *GK*. But the nice thing about the proposed scheme is that the system is potentially self-correcting. If the *GK* vouching for the spammer doesn't stop vouching, then the *GK* will likely be revoked. This may happen at many different levels, since the recipients are free to decide the level to which they want to adjust their limits to tolerate the misbehavior (including the possibility of removing the *GKs* they authorize to vouch for). In the end, the user at fault will be effectively treated as a spammer. For this reason, it is also unlikely that a legitimate *GK* will

willingly vouch for a spammer since they know what the consequence will be.

6.10 Weakness of Trust Relationships

Bilge et al. [19] showed the ease with which an attacker can automatically clone Facebook user profiles and convince large fraction of the victim's friends to become a friend of the cloned (malicious) contact. Email network can be turned into a social network/graph based on the communication pattern. In email social networks cloning a profile, which in this case is the user's email address, is not similar to cloning a Facebook profile which has all its data publicly available.

LENS identifies the notion of trust between two users with the notion of friendship in email social networks. In *LENS* when two users add each other as friends it means that both of them trust each other to be non-spammer and not necessarily some close friends. This trust is based on personal acquaintance or exchange of messages over time. The notion of friendship in an email social network is not necessarily associated to full trust with respect to the other individual's security practices, nor to his ability to wisely choose its friends. *LENS* addresses the existence of weak links (or malicious nodes) in social relations by simply reporting community members that prove to be malicious over time.

Friendship information is private and *LENS* does not exchange contacts, *CommList*, *PKList* and *VoucherList* between different *MSs*. All the information is kept on the user's *MS* just like user's very private/personal messages, email contacts and authentication (for e.g. password) information. This data is normally protected under privacy and data protection laws.

6.11 Change in Internet Infrastructure

A basic requirement for any practical system is that it does not make major changes to the Internet infrastructure (e.g. *SMTP* or other protocols). *LENS* fully complies with this requirement as it does not change existing protocols (like *SMTP*). Community formation, *GK* selection and trust management run as independent components. *LENS* based email processing (using *CommList* and *vouchers* issued by the *GK*) run as an independent daemon (like spamd for SpamAssassin) to monitor the SMTP transaction and based on the results take different actions.

Chapter 7

Evaluations

The performance of *LENS* has been evaluated using trace-driven simulations (§ 7.1, § 7.2) and a Linux based prototype implementation (§ 7.3, § 7.4). Our evaluations focused on: i) scalability, ii) effectiveness in accepting all inbound emails, iii) performance of *GK* selection, and iv) computational complexity of email processing with *LENS*.

7.1 Scalability Evaluation with OSN Data

To study the **scalability** of *LENS*, simulations have been developed based on three large scale *OSN* datasets: Facebook [73], (Google) Buzz [47] and Flickr [58]. Data samples of Facebook, Buzz and Flickr are good choices for evaluating *LENS* as they represent real online social connections. Currently, Facebook is the largest social network in the world and the number one photo sharing site on the internet. It is a "pure" social network in the sense that its primary purpose is to find and connect users. The Facebook data sample used in the evaluations consists of 3.1 million users with over 23 million edges and an average of 15.2 friends per user. The Buzz data sample consists of 2.1 million users with over 5.2 million edges and 4.5 friends on average for each user. Flickr is not a pure social network and is intended primarily for publishing, organizing and locating content. The Flickr dataset consists of 1.7 million users with over 15 million edges and an average of 18.1 friends per user. Table 7.1 presents the high-level statistics. For the evaluations 4000 users have been selected randomly from the Facebook and Flickr datasets. In the much larger Buzz data set, 1 million users are randomly chosen. In each case, the selection is restricted to users with at least 25 friends, to avoid anomalous figures due to isolated users.

For the evaluations on the *OSN* datasets, it is preferred to understand the burden imposed by *LENS* on the social network in terms of the number of GKs required, and the expected return, in terms of increased reachability via those GKs. In other words, for the results the main interest lies in: 1) reasonable number of *GKs* selected for a particular recipient and 2) reach-

ability of the recipient (i.e. maximum number of legitimate users with a minimum number of *GKs*). *Note that due to limited depth of OSN dataset the GK selection procedure has been evaluated only for stage 1.* Although the datasets contain millions of users, the average path lengths are no more than 5 hops.

Table 7.1: High-level stats of OSN datasets.

Social network data set	Facebook	Flickr	Buzz
Number of Users	3,097,165	1,715,255	2,123,421
Number of Edges	23,667,394	15,555,041	5,257,684
Average Friends	15.28	18.13	4.59
Clustering Coefficient	0.175	0.313	0.09
Avg Path Length	5.13	5.67	5.39
Average Community Size	1,587.32	4,398.44	189

7.1.1 Burden imposed: number of *GKs* required

Since each boundary user's (also *FoFs*) *MS* suggests a *GK* at each stage of *GK* selection (some *FoFs* may suggest the same *GK*, so the relationship is not strictly linear), the number of *GKs* required depends on the number of *FoFs* which in turn depends on the community size (number of friends+*FoFs*). Figures 7.1, 7.2 and 7.3 show the number of *GKs* and *FoFs* required for Facebook, Flickr and Buzz respectively. In each case, the numbers are quite modest. Facebook requires 56–871 *GKs* (<50% of community size); Flickr requires 20–400 *GKs* (<30% of community size), and Buzz requires 14–478 *GKs* (<30% of community size).

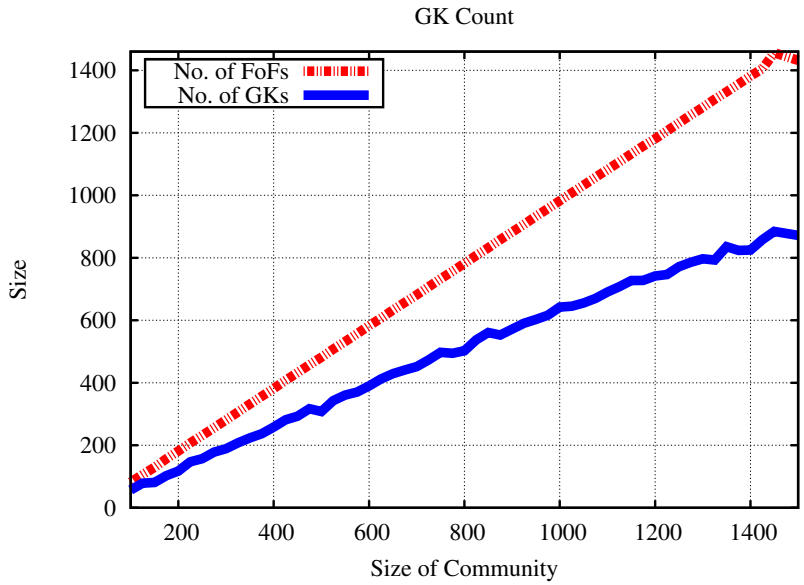

Figure 7.1: Number of *GKs* for receiving messages in Facebook dataset.

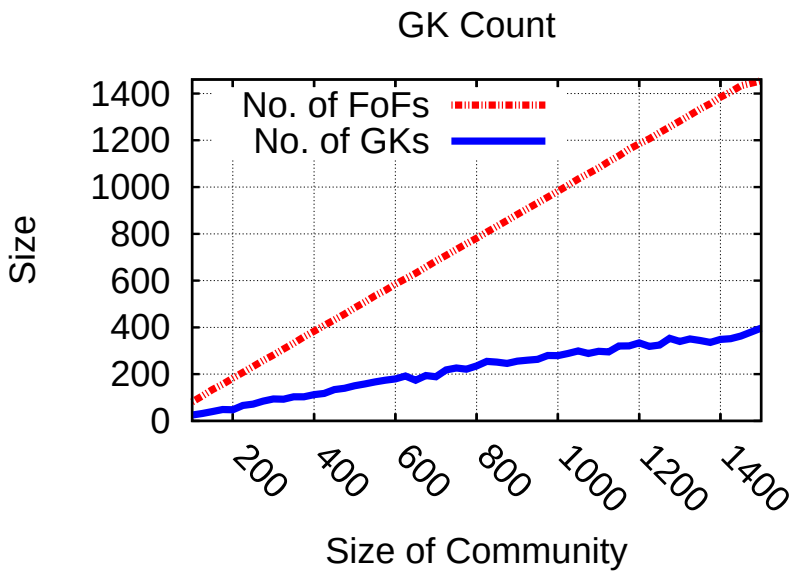

Figure 7.2: Number of *GKs* for receiving messages in Flickr dataset.

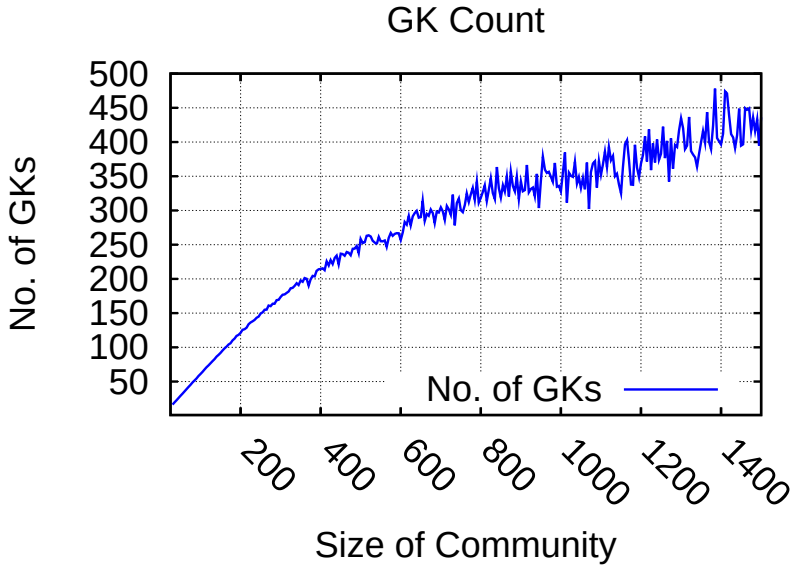

Figure 7.3: Number of *GKs* for receiving messages in (Google) Buzz dataset.

7.1.2 Expected return: increased reachability via GKs

This section presents the number of additional users that can be reached via *GKs*. A larger number of *GKs* will clearly allow more users to be reached; thus reachability has been plotted as a function of number of *GKs*. Figures 7.4 and 7.5 show the results for Facebook and Flickr respectively. The y1 axis measures absolute number of users reached (scaled to measure tens of thousands of users; i.e., a1 on the y1 axis indicates 10K users); the y2 axis gives the same number as a percentage of the total number of users in the network. Figures 7.6 and 7.7 show the absolute number (scaled to tens of thousands of users) and percentage figures for the Buzz data set.

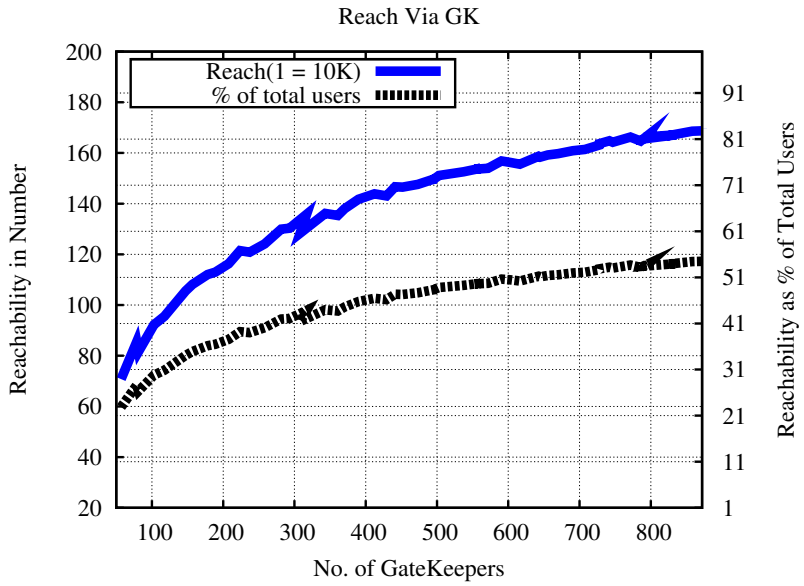

Figure 7.4: Reachability of recipient via GKs in Facebook dataset.

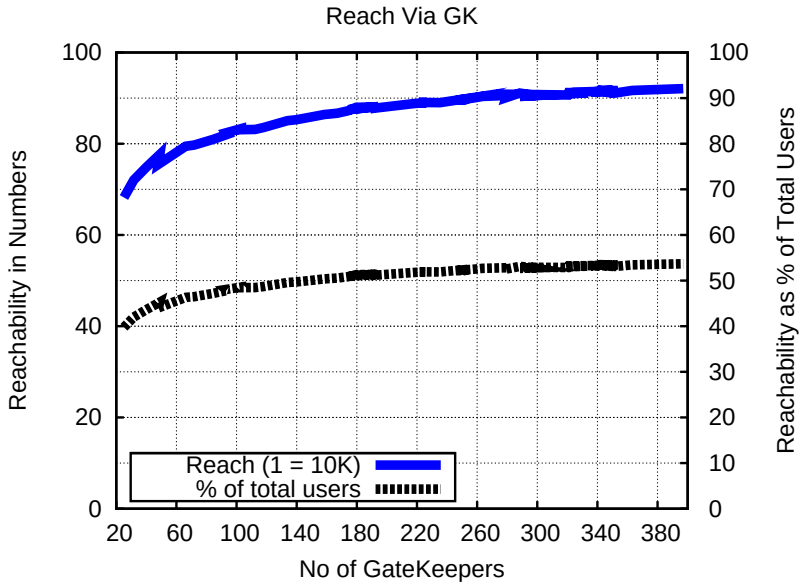

Figure 7.5: Reachability of recipient via GKs in Flickr dataset.

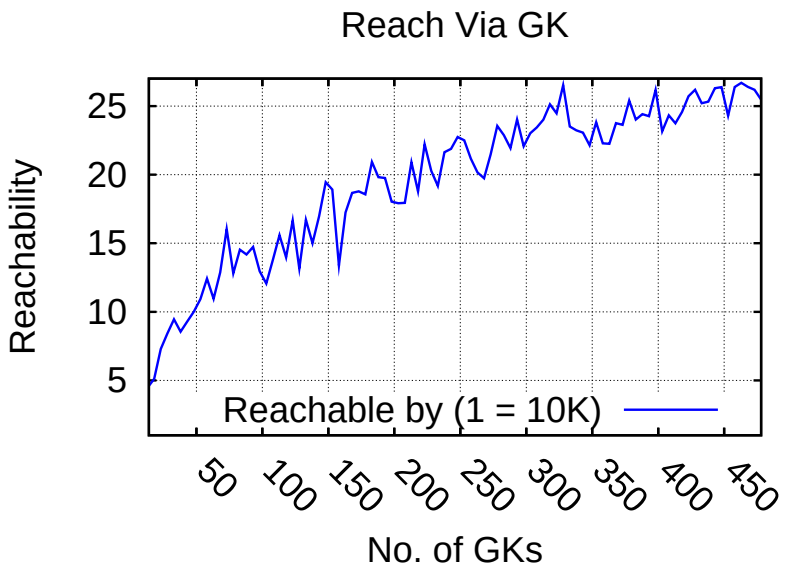

Figure 7.6: Reachability (#) of recipient via GKs in (Google) Buzz dataset.

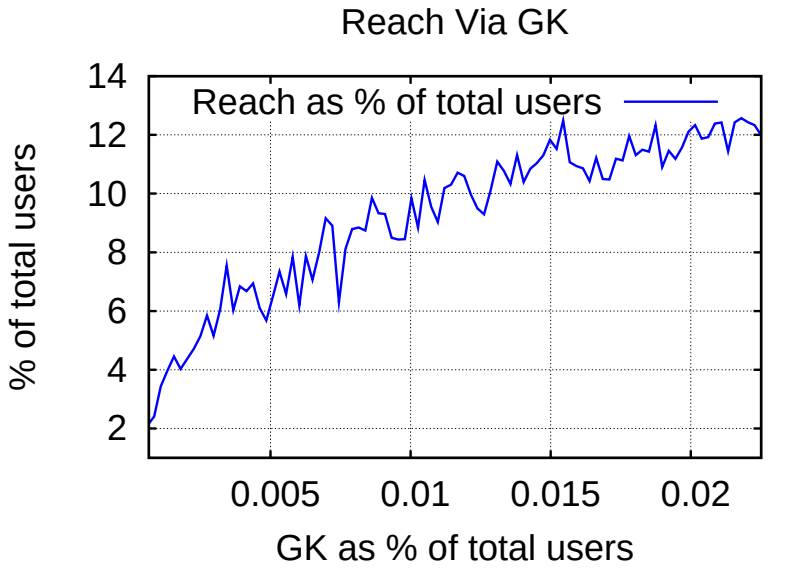

Figure 7.7: Reachability (%) of recipient via GKs in (Google) Buzz dataset.

In Facebook, with a minimum number of *GKs*, the reachability of a recipient ranges from 700K to 1.7 million which is 22% to 55% of the total network and remains over 40% most of the time. In Flickr, the reachability is between 40% to 54% of the network and mostly remains above 45%. In the much larger Buzz data set, only 0.02% of users ever get selected as *GKs*, and the maximum reachability achieved is 12.5%. It is simply because of small community size which directly affects the reach of the recipients via *GKs*. The average community size in Buzz is 189, which is very small compared to Facebook and Flickr. This seems to indicate that more than 2 rounds of *GK* selection is required to attain reasonable reachability in Buzz.

The results of these evaluations show that *LENS* is scalable in terms of number of required *GKs* and reachability. With the help of only hundreds of *GKs*, a recipient can be reliably reached by millions of users (>40% of users in Facebook and Flickr). In contrast, in *RE:*, a user is only reachable reliably from within his community (friends, *FoF*). The average reliable reachability in RE: is 0.051% and 0.25% of the network for each recipient in Facebook and Flickr respectively.

7.1.3 Altruistic commitment and security

A key requirement of the protocol is that *GK's MS*, which is unrelated to recipient's *MS* should altruistically spend resources to help it. Although *LENS* has been designed to make this as lightweight as possible (each voucher verification for a recipient requires no more than a table look up for verification), as the social network scales and more and more users are selected as *GK*, the *GK's MS* could end up being overwhelmed. To understand this impact, Buzz which has the largest data set has been studied. The results show that a total of 358K (i.e. 16%) distinct users are selected as *GKs*. More than 80% of the users selected as *GKs* have less than 300 recipients. Thus, the additional burden of altruistic processing imposed on *GK's MS* is not overwhelming for any *MS*. It is also worth mentioning that this also has a positive implication for security i.e. if and when a *GK* is compromised by a spammer, the affected number of recipients remain limited.

The results in this subsection suggest that *LENS* is scalable in terms of number of required *GKs* and reachability. With the help of only hundreds of *GKs*, a recipient can be reached by millions of users (>40% of users in Facebook and Flickr, and up to 12.5% in Buzz) and the solution can be extended to the users with even farther social distance by additional *GK* selection. In contrast, in *RE:* which handles up to FoF, a user will only be reachable reliably from its community (friends, *FoFs*). The average

reachability in *RE:* is 0.051%, 0.25% and 0.01% of the network for each recipient in Facebook, Flickr and Buzz respectively. An increase in the recipient's community size directly affects its reachability. Users having a larger community would benefit more from *LENS* than isolated and less socially connected users.

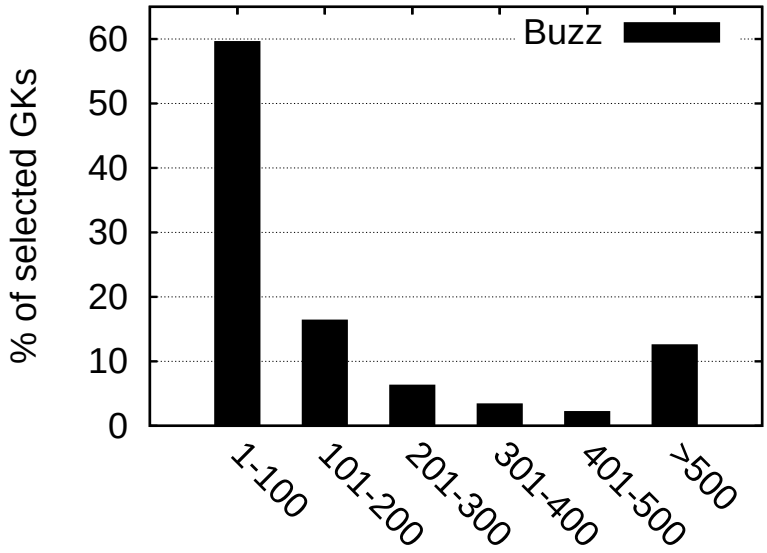

Figure 7.8: Recipients per GK in (Google) Buzz dataset.

7.2 Real Email Trace Driven Evaluation

§7.1.2 studied reachability between arbitrary sets of users. In reality, most senders can be expected to be socially close to their recipients. To evaluate the **effectiveness of *LENS*** in assuring legitimacy of senders and the ability to accept all the legitimate inbound emails, the performance of *LENS* has been studied using two real email traces. One of this trace is taken from Enron (`http://www.cs.cmu.edu/~enron/`) which is a large commercial unit, whereas the other one consists of University of Kiel's email servers log files [33]. Kiel's email server logs record the source and destination of every email from or to a student account over a period of 112 days. The dataset consists of 447,543 messages exchanged between 57,158 users. Enron's email traces are mostly of the senior management at Enron. These traces contain a total of about 1,136,760 messages exchanged between 52,747 users. Messages

with multiple recipients are counted as one message per recipient.

Following [25, 59, 65], a social network has been extracted from the email data by examining the messages sent between the users. Specifically, an edge has been created between users who sent at least three emails to each other. These edges are used to form social graphs for email users. Table 7.2 shows the high-level statistics of both datasets.

Table 7.2: High-level stats of email datasets.

Email data set	Enron	Uni-Kiel
Number of Users	52,747	57,158
Number of Edges	74,248	22,648
Messages Exchanged	1,136,760	447,543
Avg Community	167	22

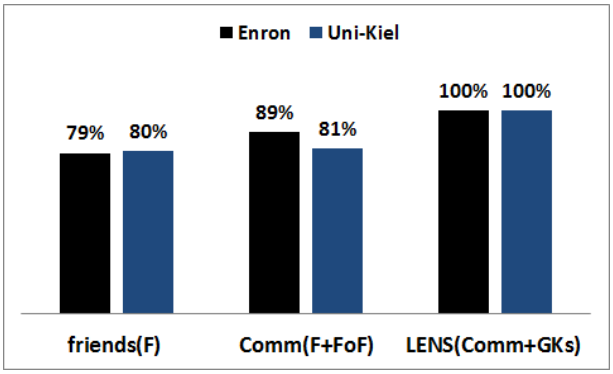

Figure 7.9: Friend, community and *GK* based filtering on Enron and Uni-Kiel email traces.

Figure 7.9 shows the acceptance of legitimate inbound email based on friends, community (friends+*FoFs*) and *GKs* in Enron and Uni-Kiel email traces. First of all, the edge information has been used to apply a friend filter on the inbound emails i.e. accept an email if it is from a friend (i.e., has sent > 3 mails). This results in the acceptance of 79% of the emails in Enron and 80% in Uni-Kiel datasets. Expanding this filter to *FoFs*, the acceptance rate increases to 89% in Enron and 81% in Uni-Kiel datasets. This is the acceptance rate of *RE:*. Finally, *GKs* are applied on the datasets using *GK* selection procedure at stages 1 and 3. This results in *100%* acceptance of the emails. Hence, the results show that *LENS* can effectively filter and accept all the legitimate inbound emails.

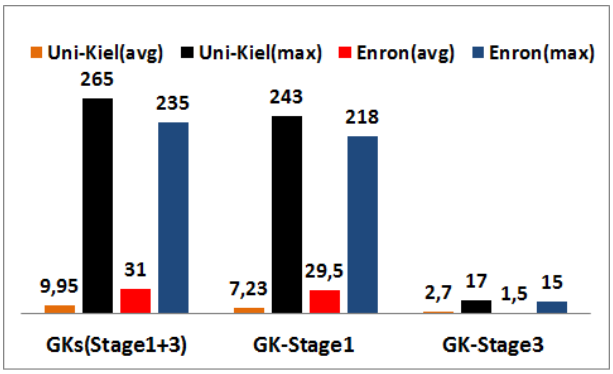

Figure 7.10: Average *GKs* in Enron and Uni-Kiel email traces.

The *number of GKs required* for Enron and Uni-Kiel datasets is quite reasonable (see figure 7.11). In Enron, the resulting number of *GKs* range between 6 to 235. The overall average of selected *GKs* is 31 per user (see figure 7.10). Out of these 31 *GKs*, 29.5 *GKs* are selected in stages 1 and only 1.5 *GKs* are selected spontaneously in stage 3. In the worst case, the *GK* count is 218 for stage 1 and 15 for stage 3.

On the other hand, in Uni-Kiel, the number of selected *GKs* is between 5 to 265. The overall average of selected *GKs* for the Uni Kiel dataset is 9.95 per user (see figure 7.10). Out of these 9.95 *GKs*, 7.23 *GKs* are selected in stage 1 and only 2.7 *GKs* are selected spontaneously in stage 3. In the worst case scenario for Uni Kiel dataset, the *GK* count is 243 for stage 1 and 17 for stage 3.

Figure 7.12 shows the reachability of a particular recipient with the help of *GKs* alone, i.e., without counting the reachability achieved by friends and *FoFs*. With the *GKs* selected above, the reachability of a recipient ranges between 8.9K to 15.8K i.e. 16.8% to 30% of the total users in Enron. In Uni-Kiel, the reachability of a recipient is between 3.3K to 13.4K i.e. 5.8% to 23.4% of the total users. In contrast, the reachability in *RE:* (depending on the community size) on average is much lower and is typically only 0.31% and 0.04% of the network for each recipient in Enron and Uni-Kiel respectively.

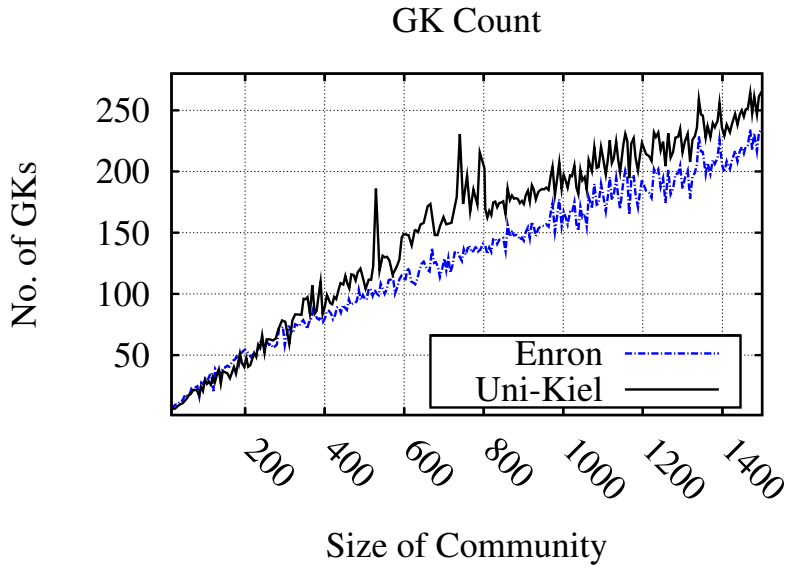

Figure 7.11: No. of *GKs* in Enron and Uni-Kiel email traces.

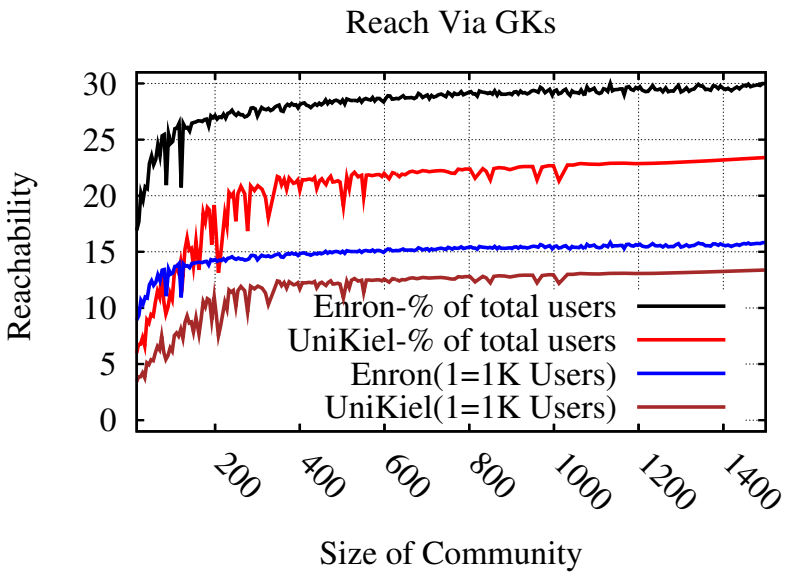

Figure 7.12: Reachability via *GKs* in Enron and Uni-Kiel email traces.

In Enron, a total of 3911 users are selected as *GKs*, which constitutes 7.4% of the total number of users. On the other hand, in Uni-Kiel, 2944 users (or 5.7% of the total number of users) are selected as *GKs*. In figure 7.13 the selected *GKs* have been distributed according to the number of recipients they were selected for. In Enron and Uni-Kiel, the distribution pattern of the number of recipients per *GK* is similar to that of Buzz. In Enron, 75% of the selected *GKs* have less than 300 recipients, whereas in Uni-Kiel, 92% of the selected *GKs* have less than 300 recipients. Thus, if a *GK* is compromised by some malicious user, a very limited proportion of recipients will be affected with spam before the system recovers.

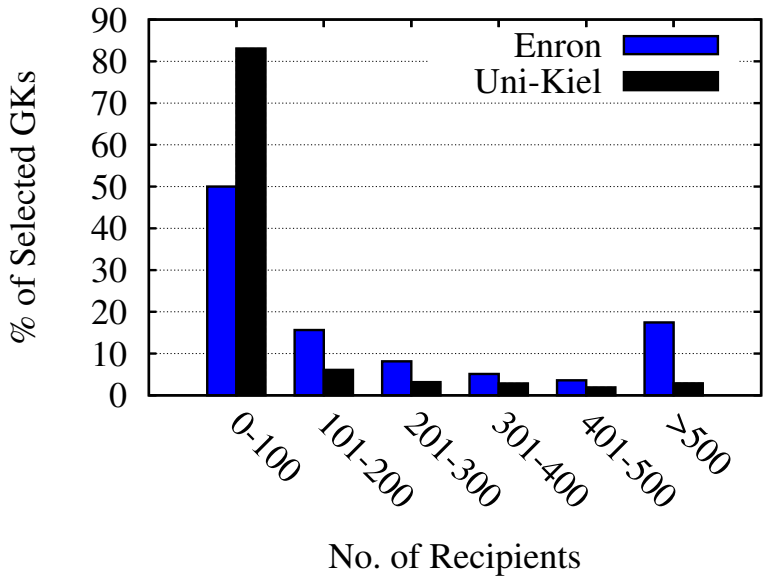

Figure 7.13: Recipients per *GK* in Enron and Uni-Kiel email traces.

These results indicate that with the help of only a few dozen *GKs*, a recipient can successfully receive all legitimate inbound emails.

7.3 Performance of *GK* Selection Protocol

Electronic mail was not designed to be time critical application and inherently the users are tolerant to reasonable delays that might occur during the transmission. Nevertheless, within the scope of this thesis, it is important that the *GK* selection does not results in huge delays and bog down the

email infrastructure. In this section, an experiment has been setup to study the latency of *GK* selection protocol, when the *MSs* are located in different countries, with the help of 20 nodes across the globe on PlanetLab. During the experiment each node sends a random *GK* selection request to one of the other 19 nodes after every 0.1 second. Figure 7.14 represents the location of the *MSs* on the map.

Figure 7.14: Location of MSs on the map

Figure 7.15 shows the average execution time of *GK* selection protocol in stage 1. During the experiment the *GK*, *recipient* and the *FoF* remained on separate nodes. The average execution time of the protocol ranged between 1.5–4.5 seconds. *GK* Selection in stage 1 involves an extra step where the *GKs* are suggested by *FoF* to the *recipient*. Thus, the execution time in stage 1 is higher than stage 3.

In stage 3, the protocol involves two parties, the *recipient's MS* and the *GK's MS*. Figure 7.16 shows the result of stage 3 execution time. The average execution time of the protocol on each node remained between 0.5–1.5 seconds with the only exception of kr node where the average execution time remained 2.6 seconds. More than 60% of the time was spent on the transmission of the message and the actual processing time remained less than 40%.

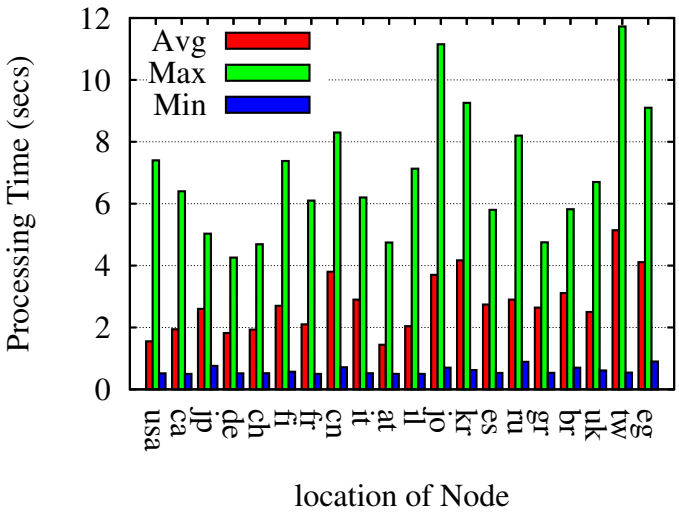

Figure 7.15: *GK* selection stage 1 (x-axis: country codes of the *MS*'s location).

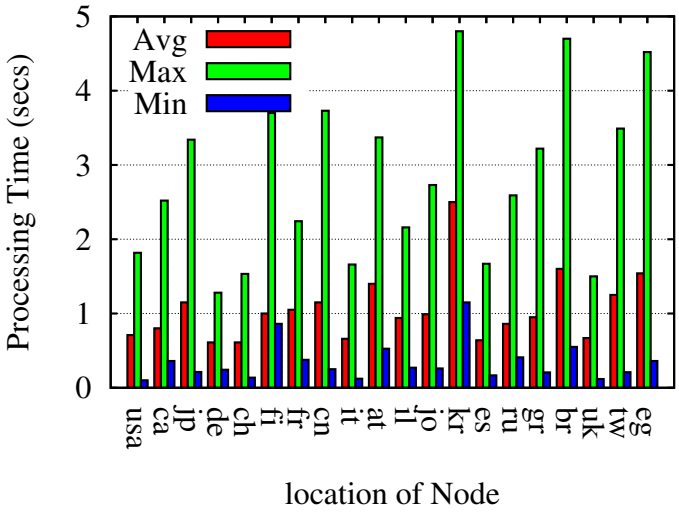

Figure 7.16: *GK* selection stage 3 (x-axis: country codes of the *MS*'s location).

Both stage 1 and stage 3 experiments finished with a success rate of

94.6% to 100% respectively. For a few nodes like kr, jo, eg and br, the success rate was not 100% due to the node getting overloaded by too many connections.

7.4 Computational Complexity of Email Processing with *LENS*

To analyze the computational complexity and system performance of email processing with *LENS*, a standard mail processing system has been augmented with *LENS* implementation, and deployed over LAN. For evaluations two experimental setups are used, which are discussed in detail below.

Experimental Setup # 1

The first experimental setup used two *MSs* running MailAvenger *SMTP* server on top of the Postfix *MTA*. The *MSs* are connected via a local area network. One of the *MS* is an Intel atom 1.6 GHz with 1 GB RAM and the other one is an Intel core2duo 2.53 GHz with 4 GB RAM. This setup is used to *measure the effect of community size, size of VoucherList and the number of GKs* on the processing emails. It is worth noting that the experiments based on setup # 1 only measure the computational overhead (in terms of time) of *LENS* on the *SMTP* transaction.

7.4.1 Effect of CommList at the recipient's MS

On receiving the `RCPT TO:` command, the recipient's MS performs a check to see if the sender is in the recipient's community, and take actions (accept/reject/pass to other filters) according to the policy. Figure 7.17, presents the results to show the overhead of community size on email processing with *LENS*. The community size varies from 0 to 10K. The effect of this variation on computation overhead on both the MSs is in a few milliseconds (ms). With the community size of 10K the overhead at one *MS* (Intel atom 1.6 GHz) is 30 ms and around 9 ms on the other MS (Intel core2duo 2.53 GHz). In Enron the average community size of any user is 167, and the email processing overhead for that remains under 3 ms for both *MSs*.

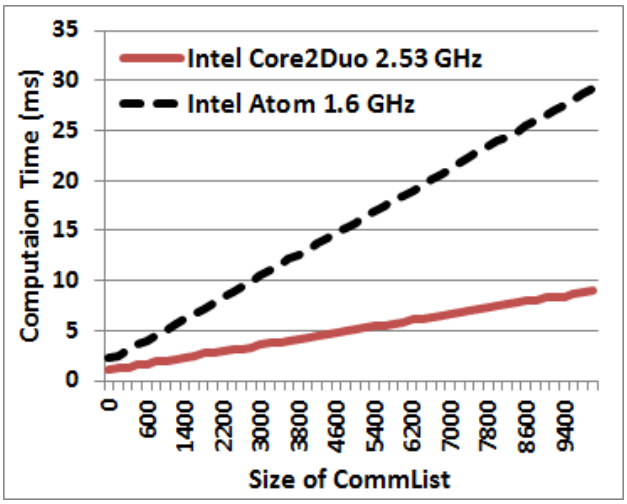

Figure 7.17: Community list lookup delays at the recipient.

7.4.2 Complexity of voucher verification and effect of PKList at the recipient's MS

The cost of checking signature has been considered substantial till now. Therefore, it is important to perform measurements to ensure that the proposed scheme, with its voucher verification does not incur more computational cost than processing spam. For the generation/verification of signature (voucher), *LENS* uses 1024 bits *RSA* and *SHA-1*. The computational complexity (in terms of time) of signature generation or verification has been tested using OpenSSL [6] speed measurement for RSA-based signatures on Intel core2duo with 2.53 GHz and 4 GB RAM. In public-key cryptography the generation and verification of signature does not have the the same complexity. It takes approximately 0.13 ms to sign and 0.07 ms to verify a voucher of 1024 bits *RSA* and *SHA-1*.

Voucher verification, when integrated with *LENS* for email processing, depends on the size of *PKList*. The size of the *PKList* of a recipient is the same as the number of its *GKs*. Figure 7.18 shows the overhead of *PKList* size on signature verification with *LENS*. The average *GK* count in Enron and Uni-Kiel is 31 and 10 respectively, and 510 in the worst case. Based on this, the size of *PKList* is varied between 1 to 1000 for the experiments. The computational overhead on both the *MSs* is only few milliseconds (ms). The overhead at one *MS* (Intel atom 1.6 GHz) is 6.7 ms and around 2.3 ms on the other (Intel core2duo 2.53 GHz) with the *PKList* size of 1000.

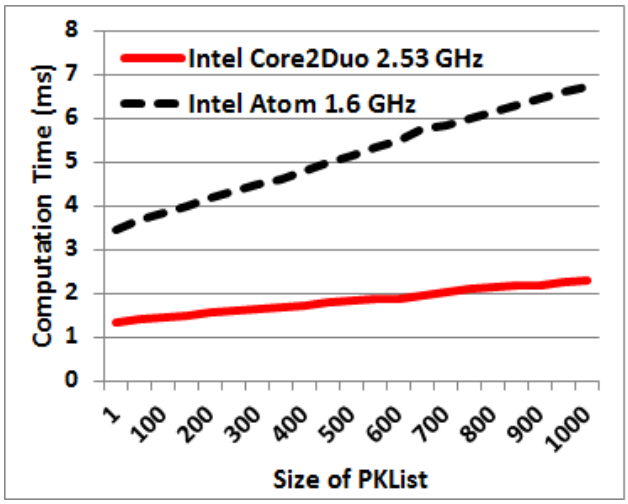

Figure 7.18: PKList list lookup delays at the recipient.

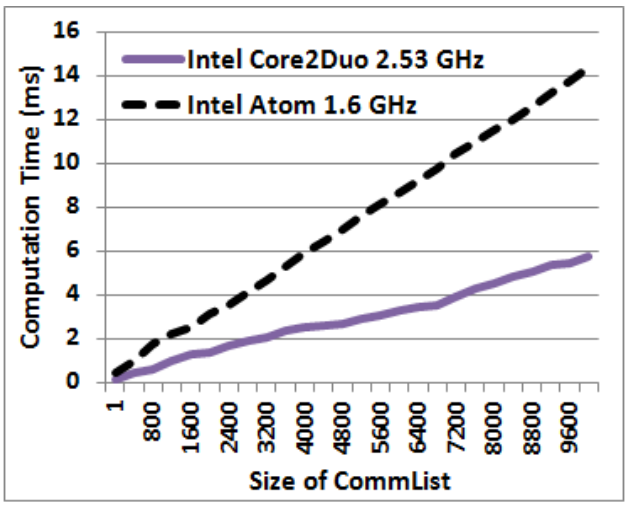

Figure 7.19: PKList list lookup delays at the sender.

7.4.3 Effect of *CommList* and *VoucherList* at the sender's MS

Before the sender's *MS* sends the `RCPT TO:` command, it performs a lookup in the *CommList* and the *VoucherList* of the sender to decide if the voucher needs to be appended or not. Figure 7.19 and 7.20 shows the overhead of the

CommList and the *VoucherList* lookups respectively. The size of *CommList* was varied from 1 to 10K and *VoucherList* from 1 to 20K. *VoucherList* lookup is performed only if the recipient is not in the community of the sender and *CommList* lookup overhead is added to the *VoucherList* lookups. In the worst case, the *CommList* lookup overhead is between 6 to 15 ms on the two MSs and the overhead of *VoucherList* lookup is between 12 to 29 ms.

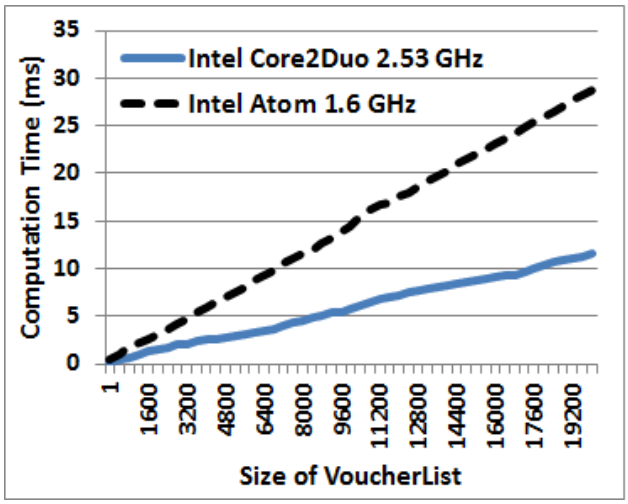

Figure 7.20: Voucher list lookup delays at the sender.

Based on the results presented in this section, it can be concluded that *LENS* is computationally efficient. *LENS* is fast in processing emails and it also scales efficiently with increasing size of the lists (*CommList*, *VoucherList* and *PKList*), and even on low end processing machines, the overhead is in milliseconds.

Experimental Setup # 2

The second experimental setup used an *SMTP* server of 2.53 GHz Intel core2duo processor, 4 GB RAM and three senders, each running on a different machine connected via *LAN* (similar to *RE*:). In this setup different experiments were conducted for 4 different scenarios to *study the impact of message size, end-to-end throughput and CPU, memory and bandwidth consumptions*. Each of these scenarios are discussed in detail below.

(S1) In scenario 1, the *SMTP* server runs postfix without any spam filter.

(S2) SpamAssassin, which is one of the most widely used content-based filter, uses a variety of mechanisms including header and text analysis, Bayesian filtering, *DNS* blacklists, and collaborative filtering databases to filter spam before it reaches the mailbox. For comparison with *LENS*, SpamAssassin is used as a content-based filter with Postfix in scenario 2.

(S3) In scenario 3, MailAvenger is used on top of postfix and *LENS* based community filtering is enabled. In this scenario the SMTP server is able to filter (accept/reject) emails at the `RCPT TO:` command based on the recipient's CommList (containing 10K entries for all the experiments).

(S4) Scenario 4 is similar to S3 with additional email filtering that is performed on the voucher's issued by the authorized *GKs* of the recipient.

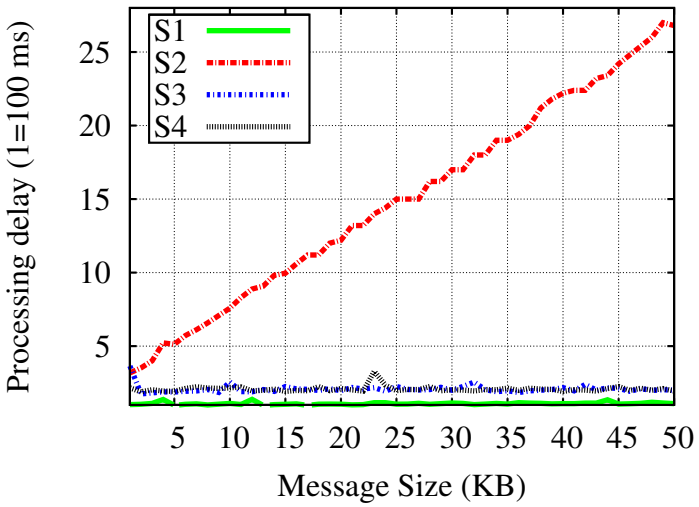

Figure 7.21: Effect of size on email processing with *LENS*.

7.4.4 Effect of message size

In order to measure the effect of message size on processing time, 50K messages are sent with varying sizes (1 KB to 50KB). Figure 7.21 shows the processing delays in all 4 scenarios. S2 exhibits a linear increase in processing time with the increase in the message size and takes 2.6 seconds to

process a message of 50KB. S1, S3 and S4 remain unaffected by the message size. S3 and S4 take slightly more time (0.2 to 0.3 secs more) than S1, which is primarily due to the additional overhead from MailAvenger and *LENS* based filtering.

Interestingly, S3 and S4 show similar processing delays, as the MS in S4 does not have to traverse the *CommList* (which is fixed at 10K entries in the experiments) and only verifies the vouchers issued by the *GK*. The results also show that the time required to process the CommList is very close to that required to verify the voucher. In short, *LENS* based email processing incurs negligible amount of additional processing delays and remains very close to S1 (no spam filtering).

7.4.5 Throughput

In order to measure end-to-end throughput of the *MS*, the senders simultaneously bombard the *MS* with enough messages to saturate it (e.g., sending 1000 messages of 8 KB each as rapidly as possible). Figure 7.22 shows the throughput of all the four scenarios. Even with the bombardment of messages S1 was able to receive 11 messages per second. This reduced down to 4.2, 4.03 and 3.2 in S3, S4 and S2 respectively.

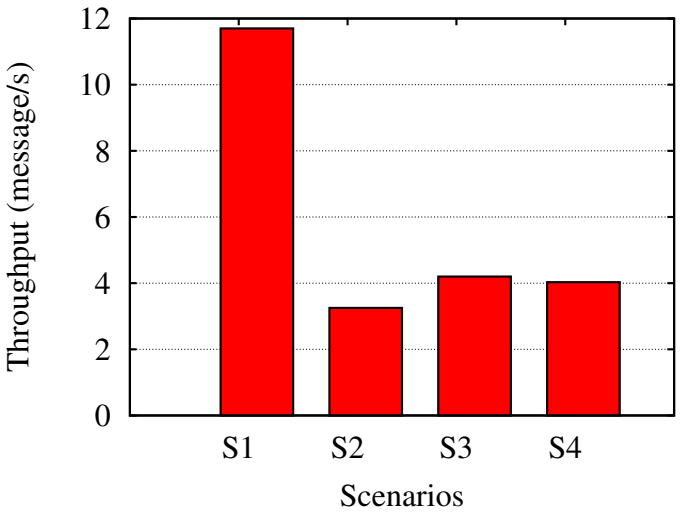

Figure 7.22: Throughput(msgs/sec).

7.4.6 CPU, memory, bandwidth and delays

In order to understand the load of emails for a large size company or university, the inbound email statistics has been retrieved from University of Göttingen for a period of one year (Feb 2010 to Jan 2011). The overall inbound emails are averaged to approximately 42K per day (see figure 7.23). Based on these figure, in the next experiment the number of inbound emails has been approximated to 50K / day with a transmission time of 8 hours. This means that fixed size (8KB) emails has been sent to the MS after every 0.6sec for 8 hours and the usage of CPU, memory, bandwidth and processing delay have been measured in all 4 scenarios.

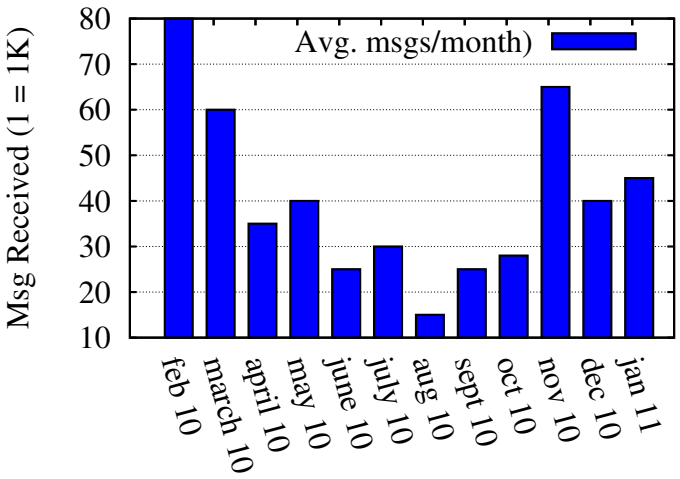

Figure 7.23: Yearly stats of inbound emails from University of Göttingen.

The CPU measurement results (figure 7.24) show S2 is very expensive in terms of CPU usage (80%-90%). This is because content-based filtering has to apply different rules and filters. In contrast, the CPU usage of S3 and S4 is similar to S1, low as 10%-12% on average.

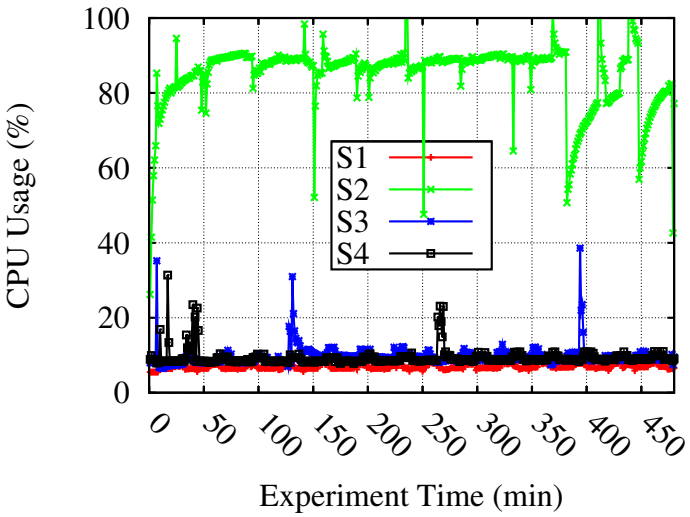

Figure 7.24: CPU usage.

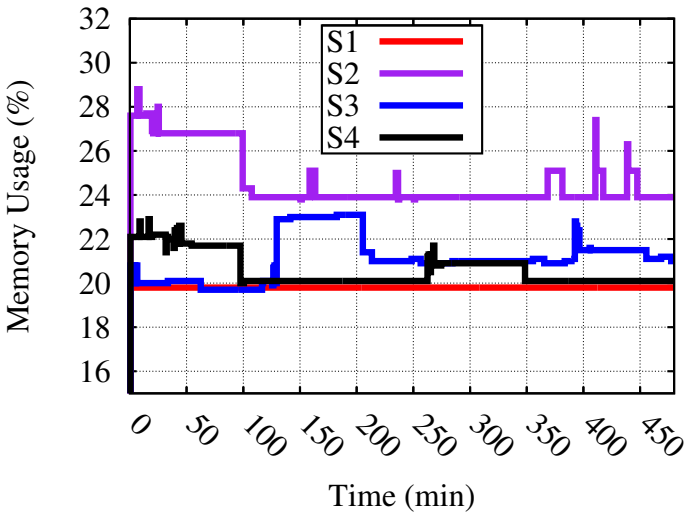

Figure 7.25: Memory usage.

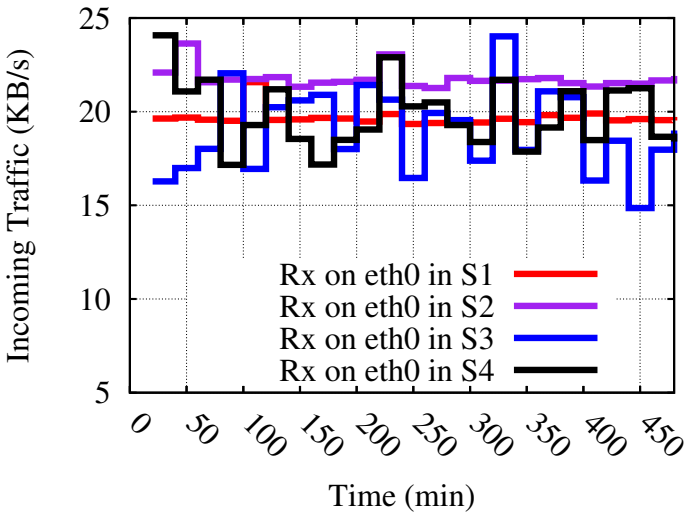

Figure 7.26: Incoming traffic.

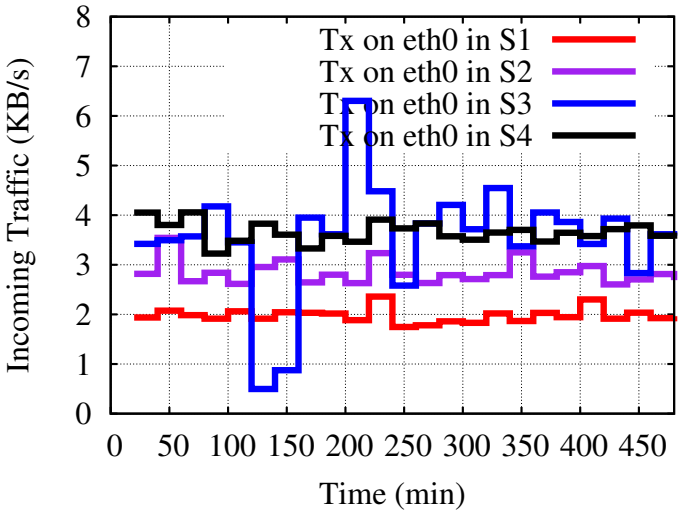

Figure 7.27: Outgoing traffic.

As shown in figure 7.25, the memory usage in S1 remained almost constant at 20%. In S3 and S4 the value remained between 20% and 23.3%. For S3 it is higher than S4 due to CommList lookup. S2 requires highest

memory (23.9%-29.1%).

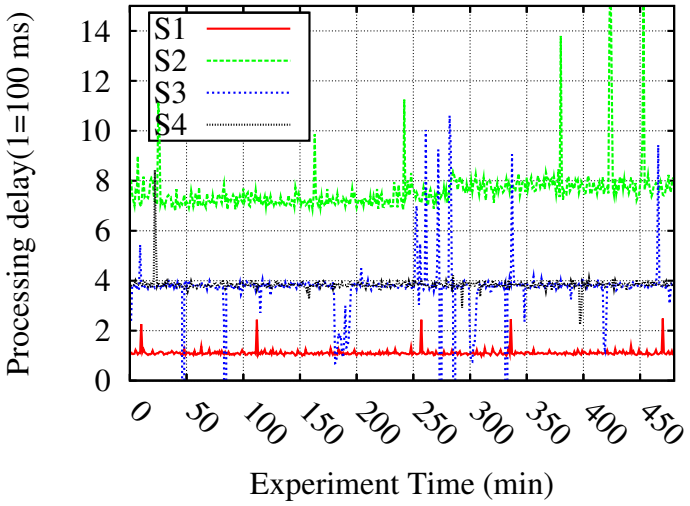

Figure 7.28: Processing delay.

Figure 7.26 and 7.27 shows that, as the message size is kept constant, there is not much difference between the 4 scenarios in terms of inbound (16-22 KB/s) and outbound (2-4 KB/s) traffic. The latter mostly constitutes the SMTP transactional traffic.

The processing delay in all scenarios (cf. figure 7.28) remained almost constant due to a fix message frequency and size. Interestingly, the processing delays in S3 and S4 are only about half of S2 and are very close to S1.

In short, *LENS* (S3 and S4) is fairly lightweight and has a performance close to the scenarios where no spam filtering is used.

7.5 Conclusion

This chapter discussed the extensive evaluations of *LENS* using three large social networks (Flickr with 1.7 million users, Facebook with 3.1 million users and Google Buzz with 2.1 million users) and two traces of email transactions (Uni-Kiel, 57.1 K users and Enron, 52.7 K users). *LENS* is able to receive *all* incoming emails in the email traces with an average of just 31 (0.06% of users) *GKs* per recipient in the Enron dataset and 10 *GKs* (0.017% of users) in the Uni-Kiel dataset. The evaluations with the social

network traces show that with the help of hundreds of *GKs*, a recipient can be possibly reached by millions of users. The solution can be scalably extended to users with larger social distances by iterative *GK* selection. *LENS* performed the worst on Google Buzz dataset, where a recipient has to choose 478 GKs on average in order to enable 12.5% of the social network to send emails to him. Whereas for Facebook with an average of 871 *GKs* a recipient can be reached by 55% of the social network. In Flickr, with an average of just 400 *GKs* a recipient is able to receive emails from 54% of the total user-base. For the Google Buzz and email traces around 80% to 92% of the selected *GKs* have less than 300 recipients. Thus, if an attacker compromise a *GK*, the affected number of recipients remains limited. *LENS* also proved to be faster in processing emails (around 2-3 orders of magnitude faster) than the most popular content-based filter i.e. SpamAssassin [10] and scales efficiently with increasing community size and *GKs* with computational overhead of a few milliseconds. The prototype implementation of *LENS* in Postfix/MailAvenger shows that the overheads imposed by additional processing are tolerably small. *LENS* consumes up to 75% less CPU and 9% less memory compared to current solutions like SpamAssassin. In short, *LENS* imposes zero overhead for the normal scenario of frequent and familiar senders, and remains lightweight for the general case.

Chapter 8

Conclusion and Future Work

The main part of this thesis presents *LENS*, a novel, easily adaptable and scalable spam prevention system with low processing overheads. Leveraging recipient's social network, *LENS* mitigates spam beyond social circles with the help of trusted users, called *GKs*. *LENS* can be deployed individually by small groups of users, and with the help of social *GKs* it is *feasible and practical* to further enhance the reliable email delivery beyond *FoFs* i.e. outside the recipient's social community.

This thesis illustrates that with only a modest number of *GKs*, each recipient is able to cover a large part of the legitimate social network, allowing most people to reach it. Furthermore, since the process of *GK* selection is performed independently by each recipient, the breakdown of individual *GKs* can only help spam those recipients who have chosen it as *GK* and do not compromise the security of most recipients. The affected recipients can curtail their damage by rotating their *GKs*. On the other hand, an attacker would have to compromise a large fraction of the set of *GKs* in order to compromise a large number of recipients.

Essentially, the protocol requires a first-time sender's (senders from outside the social circle of a particular recipient) *MS* to present a valid voucher, at the *SMTP* time, *before* sending the payload. This enables the recipient's *MS* to quickly terminate an invalid connection and stop spam at the first place from transmitting, instead of only filtering from mailboxes. Thus, the network operators can save the operating cost due to large amount of spam crossing their networks.

The key contributions of this thesis can be summarized as follows:

1. Unlike the existing social network based approaches, ***LENS* proposes a feasible and practical approach to extend spam prevention beyond the social network of a recipient**, covering all the communication scenarios for legitimate inbound emails (no legitimate email is stopped from transmission).

2. This thesis skips the cat-and-mouse game and instead of looking for

spammers, ***LENS* explores good actors in order to facilitate legitimate emails from legitimate *MSs***.

3. ***LENS* filters all the legitimate inbound emails at the *SMTP* envelop** (`RCPT TO:` command of *SMTP* transaction), before the actual transmission of email payload. This enables the recipient's *MS* to quickly terminate any invalid connection and fundamentally prevent the transmission of spam across the network.

4. This thesis also proposes *iSATS*, a new crypto-based email sender authentication mechanism. ***iSATS* is complementary to *LENS* and provide a reliable way to bind the identity of a legitimate sender to an email with stronger sender authentication than existing solutions.**

With the use of extensive evaluations, this thesis also illustrates the following.

- The evaluations on the empirical online social network datasets proves the system to be **scalable with large fraction of users and can be easily extended to millions of users**.

- The experiments on real email traces demonstrates that ***LENS* is effective in accepting all the inbound emails efficiently** and the space requirement and message overhead is also quite reasonable.

- ***LENS* remains lightweight for email processing.** The system evaluations show that *LENS* consumes up to 75% less CPU and 9% less memory as traditional solutions like SpamAssassin. The system also scales efficiently with increasing size of the lists (*CommList*, *PKList* and *VoucherList*).

- **Even for distant *MSs* the network latency for *GK* selection remains low.** With the help of 20 nodes across the globe on PlanetLab, the evaluations show that the *GK* selection protocol requires approx 0.5 – 4.5 secs.

- The system stress testing and micro-benchmarks shows that *iSATS* is computationally efficient. Even on a low end processing machine, **the overheads of processing emails with *iSATS* are in couple of milliseconds.** The *TA* also scales efficiently with burst of *SK* requests.

This thesis touches upon a new solution in the area of using email social network and *GKs* concept to establish a trust infrastructure for reliable email delivery and to stop spam from traversing across the network, but work needs to be done. Some of the directions worth investigating in future work are:

- *Automatic community formation*: In *LENS*, the community formation is a selective process involving certain human involvement to maintain high level of privacy. In order to facilitate easy and rapid adoption of the system, *LENS* should explore new mechanism to automate the community formation, while maintaining the same high level of privacy.

- *Collaborative Spam reporting*: The effectiveness of the spam reports can be increased by introducing inter-domain collaborative reporting of the spam. These reports can be stored at a central repository with easy access. This will benefit in detecting malicious users at an earlier stage.

- *LENS deployment*: This thesis integrates *LENS* with the existing email infrastructure using mutt email client, the MailAvenger SMTP daemon, and Postfix as an MTA. It would be beneficial to implement and deploy LENS in an actual commercial or academic unit to study its impact.

- *iSATS deployment*: iSATS has been deployed on a small scale. Both, the virtual and LAN environment are limited to couple of nodes. Although it is justifiable for the proof of concept and basic system performance, a real world topology with 20 plus nodes in the LAN or Planet would give more insight to system performance and its bottleneck.

Bibliography

[1] Dspam. http://dspam.nuclearelephant.com.

[2] Email address limit in webmail by providers. http://www.emailaddressmanager.com/tips/email-address-limit.html.

[3] Internet 2009 in numbers. http://royal.pingdom.com/2010/01/22/internet-2009-in-numbers/.

[4] Internet 2010 in numbers. http://royal.pingdom.com/2011/01/12/internet-2010-in-numbers/.

[5] Mail avenger. http://www.mailavenger.org/.

[6] Openssl. http://www.openssl.org/.

[7] Postfix. http://www.postfix.org/.

[8] Razor. http://razor.sourceforge.net.

[9] six botnets churning out 85 percent of all spam. http://arstechnica.com/business/news/2008/03/six-botnets-churning-out-85-percent-of-all-spam.ars.

[10] Spamassassin. http://spamassassin.apache.org/.

[11] Symantec brightmail antispam. http://www.brightmail.com/.

[12] Techrepublic. http://blogs.techrepublic.com.com/ 10things/?p=1373.

[13] Tracking the high cost of spam. http://www.redcondor.com/company/.

[14] The friend of a friend (foaf) project. http://www.foaf-project.org/, 2000.

[15] Messagelabs intelligence report: Spam intercepts timeline. http://www.messagelabs.co.uk/, July 2005.

[16] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas. Domainkeys identified mail (dkim). RFC 4871.

[17] A. Back. Hashcash-a denial of service counter-measure. http://www.hashcash.org/papers/hashcash.pdf, 2002.

[18] Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna. Four degrees of separation. *CoRR*, abs/1111.4570, 2011.

[19] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 551–560, New York, NY, USA, 2009. ACM.

[20] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001*, Lecture Notes in Computer Science, pages 213–229. Springer Berlin / Heidelberg, 2001.

[21] J. Bonneau, J. Anderson, R. Anderson, and F. Stajano. Eight friends are enough: social graph approximation via public listings. In *Proc. Second ACM EuroSys Workshop on Social Network Systems*, 2009.

[22] P.O. Boykin and V.P. Roychowdhury. Leveraging social networks to fight spam. *IEEE Computer Society*, 38(4):61–68, april 2005.

[23] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. Openpgp message format. RFC 4880, November 2007.

[24] J.C. Cha and J.H. Cheon. An identity-based signature from gap diffie-hellman groups. In *Public Key Cryptography*, Lecture Notes in Computer Science, 2002.

[25] Anurat Chapanond, Mukkai S. Krishnamoorthy, and Bülent Yener. Graph theoretic and spectral analysis of enron email data. *Comput. Math. Organ. Theory*, 11(3):265–281, October 2005.

[26] Paul-Alexandru Chirita, Jörg Diederich, and Wolfgang Nejdl. Mailrank: using ranking for spam detection. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 373–380, New York, NY, USA, 2005. ACM.

[27] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding*, Lecture Notes in Computer Science, pages 360–363. Springer Berlin / Heidelberg, 2001.

[28] Microsoft Corporation. Royalty-free sender id patent license agreement. http://download.microsoft.com/download/b/d/3/bd3b5463-c461-409c-b29f512218d3f3e6/SenderID_License-Agreement.pdf, 2004.

[29] Microsoft Corporation. Sender id framework - protecting brands and enhancing detection of spam, phishing, and zero-day exploits - a white paper. http://download.microsoft.com/download/A/6/9/A69ECA9D-6168-467E-9BEE88358B9ED595/Sender_ID_White_Paper.pdf, 2008.

[30] Dancho Danchev. Inside india's captcha solving economy. http://blogs.zdnet.com/security/?p=1835, 2008.

[31] G. Danezis and B. Wittneben. The economics of mass surveillance-and the questionable value of anonymous communications. In *Proc. of WEIS*, 2006.

[32] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 139–147, London, UK, UK, 1993. Springer-Verlag.

[33] Holger Ebel, Lutz-Ingo Mielsch, and Stefan Bornholdt. Scale-free topology of e-mail networks. *Phys. Rev. E*, 66:035103, Sep 2002.

[34] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[35] EVC. Guidelines for the issuance and management of extended validation certificates. CA/Browser Forum Version 1.2, Oct 2009.

[36] Chris Fleizach, Geoffrey M. Voelker, and Stefan Savage. Slicing spam with occam's razor. In *Proceedings of the Conference on Email and Anti-Spam (CEAS), Mountain View, California, USA*, 2007.

[37] Michael J. Freedman and Antonio Nicolosi. Efficient private techniques for verifying social proximity. In *Proceedings of 6th International workshop on Peer-To-Peer Systems, IPTPS 2007, Bellevue, WA, USA*, 2007.

[38] M. R. Garey and D. Johnson. Computers and intractability: A guide to the theory of np-completeness. W. H. Freeman, ISBN 0-7167-1045-5, p. 190, problem GT2, 1979.

[39] Scott Garriss, Michael Kaminsky, Michael J. Freedman, Brad Karp, David Mazières, and Haifeng Yu. Re: Reliable email. In *In Proc. Networked Systems Design & Implementation (NSDI), San Jose, CA*, pages 297–310, May 2006.

[40] J. Golbeck and J. Hendler. Reputation network analysis for email filtering. In *Proceedings of the Conference on Email and Anti-Spam (CEAS), Mountain View, California, USA*, 2004.

[41] Pedro H. Calais Guerra, Dorgival Guedes, Wagner Meira Jr, Cristine Hoepers, Marcelo H. P. C. Chaves, and Klaus Steding-Jessen. Spamming chains: A new way of understanding spammer behavior. In *Proc. of CEAS*, 2009.

[42] S. Hansell. Internet is losing ground in battle against spam. The New York Time, April 2003.

[43] Shuang Hao, Nadeem Ahmed Syed, Nick Feamster, Alexander G. Gray, and Sven Krasser. Detecting spammers with snare: spatio-temporal network-level automatic reputation engine. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 101–118, Berkeley, CA, USA, 2009. USENIX Association.

[44] L. G. Harbaugh. Spam-proof your in-box. PCWorld, May 2004.

[45] L. G. Harbaugh. Spam-proof your inbox. PCWorld, May 2004.

[46] Seairth Jacobs. It takes a village to stop spam. http://www.seairth.com/web/spam.html, 2004.

[47] Mohamed Ali Kaafar and Pere Manils. Why spammers should thank google? In *Proceedings of the 3rd Workshop on Social Network Systems*, SNS '10, pages 4:1–4:6, New York, NY, USA, 2010. ACM.

[48] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamalytics: an empirical analysis of spam marketing conversion. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 3–14, New York, NY, USA, 2008. ACM.

[49] J. Klensin. Simple mail transfer protocol. The Internet Society, RFC 5321, 2008.

[50] Joseph S. Kong, P. Oscar Boykin, Behnam A. Rezaei, Nima Sarshar, and Vwani P. Roychowdhury. Let your cyberalter ego share information and manage spam. *IEEE Computer*, 2006.

[51] Sailesh Kumar, Sarang Dharmapurikar, Fang Yu, Patrick Crowley, and Jonathan Turner. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 339–350, New York, NY, USA, 2006. ACM.

[52] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Félegyházi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, et al. Click trajectories: End-to-end analysis of the spam value chain. In *Proceedings of the IEEE Symposium on Security and Privacy (Oakland)*, 2011.

[53] Z. Li and H. Shen. Soap: A social network aided personalized and effective spam filter to clean your e-mail box. In *Proceedings of 30th IEEE International Conference on Computer Communications (Infocom), Shanghai, China*, April 2011.

[54] Ben Lynn. The pairing-based cryptography library. http://crypto.stanford.edu/pbc/, 2006.

[55] J. Lyon. Purported responsible address in e-mail messages. the Network Working Group, RFC 4407, April 2006.

[56] J. Lyon and M. Wong. Sender id: Authenticating e-mail. RFC 4406, april 2006.

[57] Gilad Mishne. Blocking blog spam with language model disagreement. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[58] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[59] Alan Mislove, Ansley Post, Peter Druschel, and Krishna P. Gummadi. Ostra: leveraging trust to thwart unwanted communication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design*

and Implementation, NSDI'08, pages 15–30, Berkeley, CA, USA, 2008. USENIX Association.

[60] Tatsuya Mori, Kazumichi Sato, Yousuke Takahashi, and Keisuke Ishibashi. How is e-mail sender authentication used and misused? Proceedings of the Conference on Email and Anti-Spam (CEAS), Mountain View, California, USA, 2011.

[61] M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Lett.*, 89:208701, Oct 2002.

[62] M. E. J. Newman and Juyong Park. Why social networks are different from other types of networks. *Phys. Rev. E*, Sep 2003.

[63] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.

[64] The Debian project. Deploy: Debian project unable to deploy sender id. http://www.debian.org/News/2004/20040904, 2004.

[65] J. Shetty and J. Adibi. The enron email dataset database schema and brief statistical report. Technical report, Information Sciences Institute, 2004.

[66] M. Sirivianos, Kyungbaek Kim, and Xiaowei Yang. Introducing social trust to collaborative spam mitigation. In *Proceedings of 30th IEEE International Conference on Computer Communications (Infocom), Shanghai, China*, April 2011.

[67] Symantec. 2010 annual security report, 2010.

[68] Bradley Taylor. Sender reputation in a large webmail service. In *Proceedings of the Conference on Email and Anti-Spam (CEAS), Mountain View, California, USA*, 2006.

[69] Jeffrey Travers, Stanley Milgram, Jeffrey Travers, and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32:425–443, 1969.

[70] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *CoRR*, abs/1111.4503, 2011.

[71] M. Walfish, J. D. Zamfirescu, H. Balakrishnan, D. Karger, and S. Shenker. Distributed quota enforcement for spam control. In *Proc. of NSDI*, 2006.

[72] D.J. Watts and S.H. Strogatz. Collective dynamics of small-worldnetworks. *Nature*, 393(6684):440–442, 1998.

[73] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, and Ben Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, EuroSys '09, pages 205–218, New York, NY, USA, 2009. ACM.

[74] Gregory L. Wittel and S. Felix Wu. On attacking statistical spam filters. In *Proceedings of the Conference on Email and Anti-Spam (CEAS), Mountain View, California, USA*, 2004.

[75] M. Wong and W. Schlitt. Sender policy framework (spf). RFC 4408, april 2006.

[76] M. W. Wong. Sender authentication: What to do. http://spf.pobox.com/whitepaper.pdf, July 2005.

[77] Baoning Wu and Brian D. Davison. Identifying link farm spam pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 820–829, New York, NY, USA, 2005. ACM.

Appendix A

Curriculum Vitae

Personal Data

Name:	Sufian Hameed
Date of Birth:	16 February 1985
Email:	msufianhameed@gmail.com
Address:	Robert-koch-strae 38/21 37075 – Göttingen Germany

Research Interests

- Information Security, Privacy and Trust
- Distributed Systems
- Online and Mobile Social Networks

Education

since September 2009	**PhD student in Applied Computer Science** Computer Networks Group University of Göttingen, Germany
June 2008	**MSc. Media Informatics** RWTH Aachen, Germany
August 2005	**B.S Computer Science** M.A. Jinnah University Karachi, Pakistan

Professional Experience

Georg-August-Universität, Göttingen, Germany
Research Assistant (Computer Networks Group) – since September 2009

- Research: Social Network, Anti-Spam, Email Sender Authentication
- Teaching: Seminars and Networking Lab
- Thesis Supervision: Bachelor thesis

Deutsche Telekom Labs, Berlin, Germany
Research Intern, November 2009 – March 2010

- Involved in the initial brainstorming, design and initial feasibility evaluations of a novel social network based Spam prevention system known as *LENS*.

Philips Research Labs, Aachen, Germany
Student Researcher, October 2007 April 2008

- Involved in the development of a security simulator for IEEE 802.15.4/ZigBee for mobile wireless sensor networks. Based on this simulator worked on the testing, evaluation and improvement of different security protocols for resource constrained wireless sensor nodes.

Professional Activities

- Chaired a session at 7th International Conference on Security and Cryptography (Secrypt), Rome, Italy, July 2012.
- Reviewed Papers for ICNP 2012

Teaching Experience

University of Göttingen

- Practical Course Networking Lab (SS 2011, WS 2010/2011, SS 2010)
- Seminar Internet Technology (SS 2012, WS 2011/2012, WS 2010/2011, SS 2010)

M.A. Jinnah University

- TA Computer Programming Lab (SS 2004, Fall 2004)

Publications

- **Evaluation of Human Altruism Using a DTN-based Mobile Social Network Application**, Sufian Hameed, Alexander Wolf, Konglin Zhu, Xiaoming Fu, 5th workshop on Digital Social Networks (DSN 2012) in conjunction with Gesellschaft fr Informatik (GI 2012), Braunschweig, Germany., September 2012.
- **iSATS: Leveraging Identity Based Sender Authentication for Spam Mitigation**, Sufian Hameed, Tobias Kloht, Xiaoming Fu, 7th International Conference on Security and Cryptography (Secrypt), Rome, Italy, July 2012.
- **LENS: Leveraging Social Networking and Trust to Prevent Spam Transmission**, Sufian Hameed, Xiaoming Fu, Pan Hui and Nishanth Sastry, The Second IEEE Workshop on Security and Trust in the Future Internet (FIST'11), Co-located with IEEE ICNP 2011, Vancouver, BC Canada, October 2011.
- **Leveraging Social Networking and trust to prevent Spam transmission**, Sufian Hameed, Pan Hui, Xiaoming Fu, Nishanth Sastry, 5th ACM EuroSys Doctoral Workshop (EuroDW 2011), Salzburg, Austria, April 2011.

Thesis Supervised

- Bachelor Thesis: Evaluation of Human Altruism with DTN based data forwarding, Alexander Wolf, Center of Computational Sciences, University of Göttingen, Germany, ISSN 1612-6793, No. ZAI-BSC-2012-03, February 2012.
- Bachelor Thesis: Leveraging Identity-based Authentication for Email Sender Authentication, Tobias Kloht, Center of Computational Sciences, University of Göttingen, Germany, ISSN 1612-6793, No. ZAI-BSC-2011-18, October 2011.

Patent

Title:	**Method and system for forming a network for spam free e-mail communication.**
Pub. No.:	WO/2011/131228
Application No.:	PCT/EP2010/055183
Pub. Date:	27.10.2011

Awards / Honors

- EuroSys 2011 Travel Grant
- Merit based Master's scholarship, RWTH Aachen
- 1st class distinction (summa cum laude)
- Merit based scholarship for complete bachelor studies.