Improving Email Trustworthiness through Peer-to-peer Sender Authentication

Vivek Pathak and Liviu Iftode
Division of Computer and Information Sciences
Rutgers, the State University of New Jersey
110 Frelinghuysen Road
Piscataway, NJ 08854-8019
USA
{vpathak,iftode}@cs.rutgers.edu

Abstract

The increasing use of email for phishing and unsolicited marketing has reduced the trustworthiness of email as a communication medium. Sender authentication is a known defense against these attacks. The existing proposals for sender authentication either require infrastructural support or break compatibility with existing email infrastructure. We propose, implement, and evaluate Peer-to-Peer Sender Authentication, an incrementally deployable and backward compatible sender authentication mechanism for email. The mechanism is implemented entirely in the mail client in accordance with the end-to-end principle. Sender authentication is achieved by executing our Byzantine fault tolerant public key authentication protocol [19] as an overlay on the mail transport protocol. Our sender authentication solution requires honest majority instead of trust infrastructure or human input for correctness. We evaluate the authentication overhead by running an instrumented Thunderbird mail client [18] with synthetic data showing an increased latency of about 200*ms for the user. Usability of authentication in real life is studied with two anonymized email traces. The results show that about* 40% *of the peers can be authenticated over the* 92 *day trace period.*

1 Introduction

Electronic mail is one of the most popular applications on the Internet. Unlike traditional mail that can be signed by hand, electronic mail does not have any built in authentication mechanism. In particular, the absence of sender authentication makes it possible to spoof sender identity. It is also possible to modify message contents en-route because messages do not carry digital signatures, which could provide message authentication. The lack of sender authentication and message authentication limits the effectiveness and trustworthiness of electronic mail. It is non trivial to determine the true identity of the sender because messages could be spoofed, i.e. appear to be from a different sender than the real sender. The low cost of sending electronic mail coupled with ease of spoofing has led to a flood of spam on the Internet. Having sender authentication would not only contain spoofing, but also enable tackling the spam problem by using authenticated sender identities to classify messages as trusted or otherwise. Similarly, message authentication would increase trustworthiness of electronic mail making it more effective for personal and business use. These motivations make sender authentication and message authentication important enhancements to electronic mail.

While the original electronic mail specification [5, 17] does not address authentication, the S/MIME enhancements [21, 22] have added support for message authentication. Message authentication in S/MIME depends on sender authentication, which is provided by an external public key infrastructure (PKI). This works well in an organizational setting where a central trusted party can certify public keys associated with all the mail addresses. However, the centralized trust model becomes unsuitable for communications across organizational boundaries or for private communication through free email systems. Since the email user base is decentralized with peers belonging to different logical trust domains, the authentication infrastructure should be decentralized too. This requirement is not addressed by the S/MIME standard.

A popular security add-on for electronic mail is Pretty Good Privacy, commonly known as PGP [30]. It allows users to authenticate public keys of other users in a peer-to-peer manner. Human judgment of trustworthiness guides the authentication decisions. Authenticated public keys can provide both sender authentication and message authentication through digital signatures [20]. Although

PGP has been freely available for over a decade, it is yet to be adopted by a majority of mail users. A number of factors contribute to its limited use. Firstly, it requires a significant level of sophistication to evaluate the trustworthiness of peers. It can be argued that common users are either unwilling or unable to make these decisions [28]. Secondly, even if the users are willing to put a one time effort to manage their trusted peer groups, known as key rings in PGP terminology, they may be unwilling to invest this effort on a regular basis. The vast user base of electronic mail views it as a plug-and-play service with free email accounts available from a number of providers. Therefore, a suitable authentication method must be autonomous, i.e., able to run with minimal or no human input. This requirement is not addressed by PGP.

Therefore, we believe that a widely acceptable electronic mail authentication solution must support the following requirements:

1. Operate without dependencies on centralized third parties for authentication decisions.

2. Provide autonomous operation with minimal human intervention.

Fortunately, the authentication requirements for electronic mail are less strict than those for authorization and access control [1]. Eventual authentication is acceptable in the mail environment because of its asynchronous communication pattern. Eventual authentication is also useful because it can overcome the false positives of spam filters and provide confidence in the received message, albeit with a delay. On the other hand, while electronic mail communication crosses organizational and administrative boundaries, it typically occurs within groups of collaborating individuals. Therefore, it is reasonable to assume that the groups of collaborating individuals have honest majority.

1.1 Our solution

In this paper, we propose to authenticate electronic mail by authenticating the public keys of senders through our Byzantine fault tolerant public key authentication protocol proposed in [19]. The public key authentication protocol runs as an overlay on the mail transfer protocol [11]. Electronic mails are digitally signed to authenticate senders and message content [15].

Our public key authentication protocol provides eventual authentication. This means that users may receive digitally signed messages from peers whose public keys are yet to be authenticated. While eventual authentication of sender public key would authenticate both the sender identity and the message content, there exists a period before eventual authentication where the public key authenticity is undecided. This is different from a false negative where the public key of an honest peer can not be authenticated. We note that while authentication can be delayed, there are no false negatives in public key authentication as shown in [19]. Similarly, there are no false positives under the honest majority assumption.

Since the underlying public key authentication protocol is autonomous and decentralized, mail authentication inherits these characteristics. Authentication is supported without additional infrastructure or human input. Therefore, our electronic mail authentication scheme is compatible with the usability requirements described above. Our solution allows incremental deployment and preserves compatibility with existing email infrastructure because the authentication overlay is implemented through optional extension fields supported by the mail transport protocol. In summary, this paper makes the following main contributions:

- We implement peer-to-peer sender authentication for electronic mail[2]. Our solution is automatic, byzantine fault tolerant, eventually correct, incrementally deployable, backward compatible with the existing electronic mail infrastructure, and does not use trusted third parties.

- Performance of the proposed solution is investigated through micro-benchmarks, simulation on an industrial and an academic mail trace, and live experimentation on an instrumented mail authentication prototype.

The remainder of this paper is organized as follows: Section 2 discusses email usage patterns and their impact on various sender authentication approaches. Section 3 provides a higher level summary of the public key authentication protocol. Section 4 investigates design issues and alternatives. Implementation of the authentication plugin for Thunderbird mail client is described in Section 5. The experimental studies to investigate authentication cost and performance are described in Section 6. Comparison with alternatives is done in Section 7, and Section 8 concludes the paper.

[1] We characterize authentication as eventual if it is achieved with a finite delay, which depends on the communication frequencies of the participants in the authentication protocol.

[2] We note that the design of the public key authentication protocol proposed in [19] is not the contribution of this paper. Instead, this paper applies the protocol to email that lacks autonomous sender authentication as noted earlier in this section. Minor enhancements that are required to operate in the connectionless email environment are noted in Section 4.

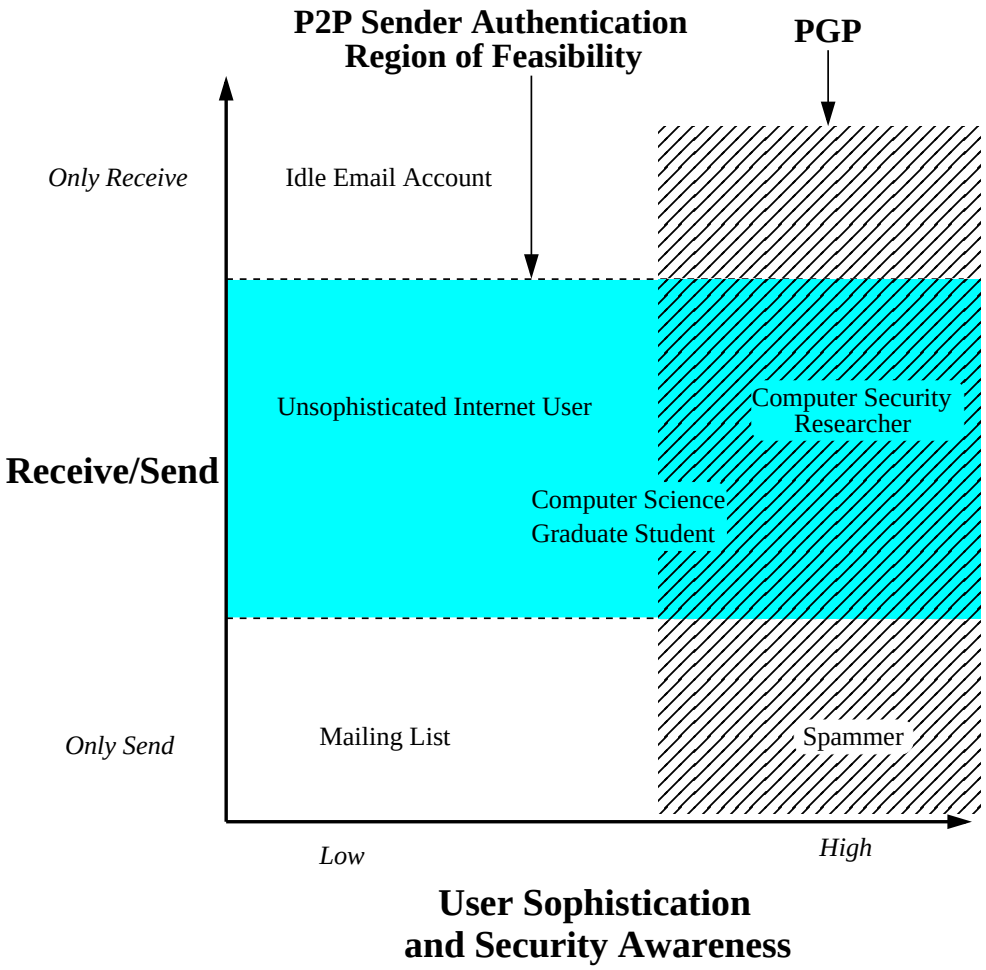

Figure 1: Peer to peer Sender Authentication requires both sending and receiving activity for feasibility. It is useful even for unsophisticated Email users.

2 Usage pattern and solution space

The underlying authentication protocol requires message exchange for progress. This requirement maps to sending and receiving emails because the authentication protocol operates as an overlay on email. As shown in Figure 1, email accounts used for both sending and receiving emails can take advantage of the authentication protocol. However, the overlay functions effectively if the communication pattern between any pair of peers is unidirectional. Note that the established methods of sender authentication do not appear to have this drawback. PGP keys can be authenticated with face to face communication. PKI can be setup administratively without requiring any action on the part of its users. However, both of these methods use the secure side channel of acquaintance or administratiion to achieve authentication. Therefore, they too require two way communication although not through the email medium. Our solution requires that email accounts be used for both sending and receiving emails. This usage pattern requirement exactly matches the basic purpose of having email accounts. Peer to peer sender authentication is therefore useful for practical applications.

Sophistication and security awareness of email users is the other dimension of email usage. Users may either be aware or unaware of the security risks and mitigating mechanisms. High security awareness means that users are able to verify and understand digital signatures, public key certificates, web of trust, public key infrastructure, etc. Unlike the established methods of sender authentication, peer-to-peer sender authentication is useful for all types of email users as shown in Figure 1. PGP allows public key authentication through direct human contact, which is stronger than the honest majority assumption used in this work. We note that PGP has solved the sender authentication problem for the (relatively) small number

	Activity Profile		User Sophistication		Organizational Use	
	Unidirectional	Send & Receive	Low	High	No	Yes
PGP	Yes	Yes	*No*	Yes	Yes	Yes
PKI/SMIME	Yes	Yes	Yes	Yes	*No*	Yes
P2P Sender Authentication	*No*	Yes	Yes	Yes	Yes	Yes

Table 1: Usage Patterns and Usability of Sender Authentication Techniques.

of sophisticated users who are willing to take the effort of authenticating public keys through direct offline contact [3] and maintaining the web of trust by selecting and certifying trustworthy peers. However, there remains a vast majority of lay Internet users who are neither sophisticated enough to use PGP's web of trust, nor patient enough to apply the effort required for authenticating public keys of new contacts. We address the needs of these users. The effectiveness of sender authentication methods is summarized in Table 1. Our solution is usable regardless of user sophistication or organizational support.

3 Authentication protocol

We denote the secure association of public keys to email addresses as "public key authentication" in this paper. This section provides an overview of our previously proposed byzantine fault tolerant public key authentication protocol. This section is included for convenience only, the reader is requested to refer [19] for details.

Participation in the protocol authenticates the self generated public keys of peers. The underlying authentication protocol assumes that participating peers are identified by their network addresses, which are email addresses in the context of this paper. The asynchronous network connecting the peers is required to provide delivery failure notifications for non-existent destinations. Also, the network should support eventual delivery on retransmissions and not become permanently partitioned[4]. We note that the email network satisfies these requirements [11].

3.1 Threat model

Adversaries mounting passive attacks are allowed to overhear all the communication between peers. The active attacks are restricted compared to the classical "network is the adversary" model as follows: The active adversaries have unlimited spoofing power, i.e., they can inject arbitrary messages into the network. However, they have limited power to prevent message delivery. In particular, for the authentication protocol to operate at a peer P, it should be impossible to prevent (eventual) message delivery for more than a fraction ϕ of P's peers. We note that since email servers are widely distributed, a practical value of ϕ would be zero for general email communication over the Internet.

[3]For example, PGP keys can be authenticated through direct contact in social gatherings known as "PGP key signing parties".

[4]Assuming that temporary failures are eventually repaired.

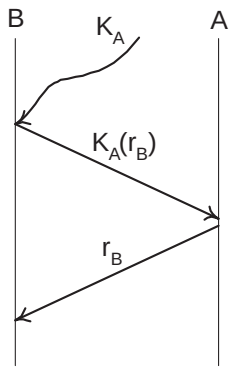

Figure 2: Challenge response protocol. B uses the nonce r_b to authenticate the public key K_A of A in absence of man in the middle attack.

Peers participating in the authentication protocol can be honest, malicious, or faulty. The protocol does not distinguish the latter two cases, but provides public key authentication service to the honest peers. The protocol correctly authenticates the public keys of honest peers if fewer than t of the n participating peers are malicious or faulty, where:

$$t = \frac{1 - 6\phi}{3} n \tag{1}$$

3.2 Authentication sketch

Challenge response protocols can authenticate public keys correctly in absence of "man-in-the-middle" attacks [7, 16, 14]. The challenge response protocol shown in Figure 2 authenticates A's public key to B. Now consider the same protocol from another peer C, whose public key has already been authenticated. C can construct an "authentication vote" for K_A by digitally signing the challenge and sharing the (digitally signed) response received from A. This authentication vote from C is correct if it is honest and there are no man-in-the-middle attacks.

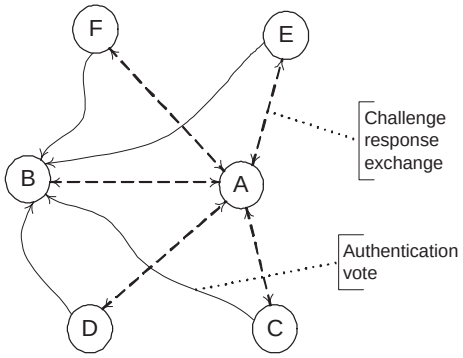

Figure 3: Distributed authentication involves multiple challenge response exchanges followed by gathering authentication votes.

The attack model permits a limited number of man-in-the-middle attacks. Also, a limited number of peers can be malicious or faulty. Therefore, public keys can be authenticated by distributed authentication where authentication votes are received from multiple peers as shown in Figure 3. Here B authenticates the public key of A by receiving authentication votes from peers C, D, E, and F. Peers construct authentication votes from their challenge response exchanges with A. The authentication votes are correct, and agree if the participants are honest and there are no man-in-the-middle attacks.

Disagreeing authentication votes imply that some of the peers are malicious, faulty, or are under a man-in-the-middle attack. Since any peer could be malicious, faulty or under the attack, a byzantine agreement [13] has to be done on the authenticity of K_A. Byzantine agreement is done by peers sending digitally signed challenge and response messages seen by them to all the participating peers. Digitally signed messages ensure that malicious behavior of any peer is provable to all the peers. Each honest peer determines the correctness of a given challenge response exchange by majority on the n forwarded authentication votes. If the trusted group with t malicious or faulty peers satisfies equation 1, then malicious peers and paths under the man-in-the-middle attack are detected and eliminated from trusted groups.

3.3 Authentication in large peer groups

The authentication protocol has a linear messaging cost in peer group size without byzantine agreement. The messaging cost becomes quadratic if byzantine agreement is required for handling faults or attacks. In order to scale the protocol to practical scenarios, the protocol is executed only on a subset of peers. This subset is composed of three groups: the trusted peers, the probationary peers, and the untrusted peers.

Public keys of trusted peers are already authenticated. The public keys of probationary peers are in the process of being authenticated, and the untrusted group is composed of peers that are known to be malicious or faulty from previous byzantine agreements. The trusted group is periodically pruned to eliminate older trusted peers through the "group migration" process. New peers are also authenticated and added to trusted groups according to the authentication needs. This process of authentication is bootstrapped with a user selected bootstrapping trusted group.

Since overtly malicious peers are detected and eliminated by byzantine agreement, covertly malicious peers can be assimilated in the trusted groups with a probability no larger than that of random selection. The universe of peers satisfies equation 1 by assumption, therefore the likelihood of having honest majority trusted groups is greater than $\frac{1}{2}$. A public key authenticated over k different trusted groups is correct with a probability greater than $2^k : 1$. This is smaller than the chance of randomly guessing an encryption key even at small values of k. Therefore, the group migration process eventually authenticates the public keys of peers.

The authentication protocol messages can be propagated through an epidemic algorithm. This eliminates the direct connectivity requirement and improves the messaging cost of the protocol to $\log n$ for a trusted group of size n. Signed messages can be safely stored and delivered through intermediate peers. As with other epidemic algorithms, this "public key infection" protocol uses timestamps to determine message delivery [6]. This knowledge is used for trimming delivered messages from peer caches.

4 Email authentication

This section discusses the design of the authentication plugin for electronic mail client. The application design has three basic goals. It should have backward compatibility with existing mail infrastructure, satisfy the constraints of the authentication protocol, and provide authentication results in a reasonable time. These goals are met by the design as explained in this section.

4.1 Overlay considerations

We use the SMTP extension fields to create an overlay for the public key authentication protocol. This is done to maintain compatibility with existing mail infrastructure. Mail systems are required to ignore user defined fields that cannot be processed [17]. Therefore, the mail messages sent by the authentication enabled mail clients would contain authentication protocol messages that are processed only by authentication enabled clients. Other mail clients would ignore the extension fields and deliver the under-

lying email to the recipient. Running the authentication protocol as an overlay over the email network introduces performance limitations and design constraints. This section investigates these issues in order to choose authentication protocol parameters that are practical in the email environment.

4.1.1 Trusted group size limits

Byzantine fault tolerant public key authentication protocol was bench-marked [19] through a simulation that investigates the cost of public key authentication. The cost of the protocol depends on various controllable parameters like bootstrapping group size, trusted group size, probationary group size, and the rate of authentication of new peers. These parameters should be selected in order to match the available computational and messaging power in typical electronic mail systems with the requirements of the protocol.

The authentication protocol can operate as an overlay above the mail transport by using the extension fields defined in SMTP. This is in line with many anti-spam implementations. However, SMTP mail transfer agents impose a limit on maximum header size. This is done to avoid denial of service attacks. For example, `sendmail`, a popular UNIX mail transfer agent, supports the maximum header size of 32 KB. This limits the maximum authentication payload that can be attached to a single message. Since the authentication protocol requires increasing amounts of messaging overhead with increasing trust group size, the maximum group size that can be supported in the overlay is limited. A high compression ratio can be achieved on the authentication messages because they are are serialized in XML format. Using a public key size of 1Kb, and ZLIB library for compression, we tested the final header load for different authentication message payloads. A moderate value of 300 authentication messages was chosen in order to impose less than 10KB overhead on the mail header.

Authentication message load depends on the size of trusted group size and the rate of discovery of new peers. Therefore, the budget of 300 authentication protocol messages per email affects the maximum size of trusted group that can be maintained. Getting hold of mailbox statistics is challenging because of privacy issues. Therefore, we gathered statistics of unique mail addresses and number of messages from the mailboxes of a few colleagues. The results indicate that about average 20% of the messages are sent to, or received from new peers that need to be authenticated. Applying this ratio to the limit of 300 authentication messages per email, we can afford 1500 authentication messages per un-authenticated peer. Our previous simulation results in [19] indicate a limit of 75 trusted peers for this messaging cost. Designing this limit into the system ensures that we can successfully bootstrap the system at any given point of time. We also note that while the cost on trusted group is independent of group size, the peers to be authenticated do incur an increasing messaging cost with group size. Since the messaging cost on the probationary peer is same as the bootstrapping cost, the proposed group size limit is also appropriate for continuous operation.

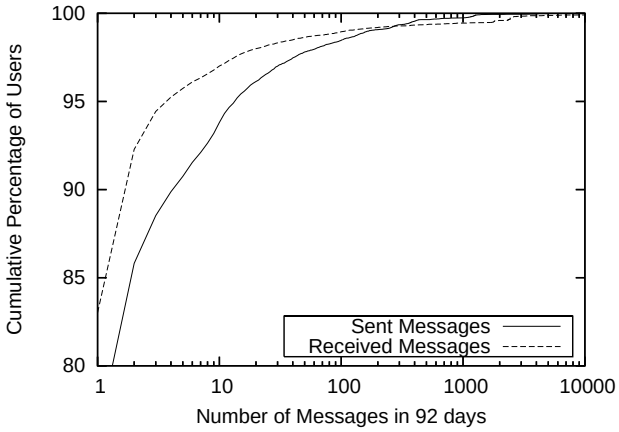

Figure 4: Activity profile of user accounts.

4.1.2 Bootstrapping trusted groups

Byzantine fault tolerant authentication provides a specification of the protocol to follow for authentication but leaves the selection of bootstrapping trust group as an application choice. The bootstrapping group naturally maps to email addresses commonly found in the mail box. The bootstrapping group is determined locally by analyzing the communication patterns in the user mail folders[5].

To determine a meaningful heuristic for generating bootstrapping trust groups, we analyzed the email communication patterns available from the anonymized University email trace (described in detail in Section 6). Figure 4 shows the cumulative distribution of number of user accounts with respect to email messages sent or received over a 92 day period. We find that a large number of user accounts are idle with minimal sending and receiving activity. Using the distribution, we cutoff accounts that do not have at least 3 outgoing messages and at least 9 incoming messages over the period of the study. This reduces the number of user accounts in the study from $27,623$ to 715. This active subset of user accounts is analyzed

[5]We note that bootstrapping group members could also be added through user choice. However, the impact of this choice cannot be analyzed without large scale deployment. Therefore, we generate the bootstrapping group using the communication pattern in the mail box.

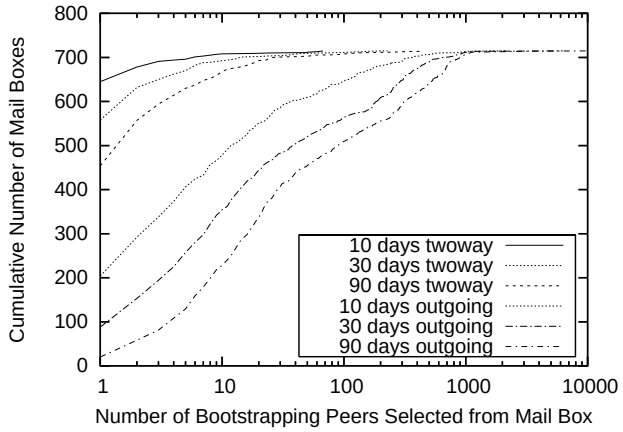

Figure 5: Selection of bootstrapping groups.

against two possible heuristics for generating bootstrapping trust groups:

- **Outgoing:** Select addresses that are destinations of outgoing email messages.
- **Two-way:** Select addresses that are either source or destination of email messages received or sent respectively.

The selection heuristics are applied to the mail trace by considering the first 10, 30, and 90 days of the trace. Using the initial subset of the trace is desirable because future communication patterns will not be available in real life. The size of the bootstrapping group for each mail box is calculated using the given heuristic and time window from the mail trace. The cumulative number of mailboxes having more than a given number of bootstrapping peers is plotted in Figure 5. It can be observed that one way communication is quite common in email as shown by the gap between the two heuristics. In order to have a frequently communicating subset of users, we apply the 30-day two-way heuristic on the 92 day mail trace. This results in a set of 53 peers that have at least 4 peers in their bootstrapping group. This subset of active users is chosen as the experimental base.

4.2 Handling the email communication model

A characteristic feature of electronic mail is that the sender and receiver do not need to be connected at the same time. This offline nature of email requires changes to the epidemic public key infection algorithm that works with anti-entropy sessions between peers. Anti-entropy sessions determine the set of messages to be sent by checking the time-stamp vectors at the receiver [12, 9, 6, 19]. We modify this approach to compute the set of messages based on stale timestamps from a previous receive event. This approach effectively splits the anti-entropy exchange into two one sided sends. It increases the cache usage because stale timestamps require sending a larger number of messages. Since the receiving peers would silently ignore duplicate messages, the modified connectionless epidemic algorithm maintains the functionality of public key infection.

Email supports one to many communication. This is handled by computing the set of messages to be sent based on all the receivers. The epidemic algorithm sends messages that may have not been received by any of the receivers in order to guarantee eventual delivery.

4.3 Trust and multiple identities

As the email client is in first hand contact with mail users, we allow manual overrides over protocol decisions. It can be observed that in absence of the autonomous authentication protocol, such a system would be equivalent to PGP in terms of the trust model. The authentication protocol periodically re-authenticates trusted peers through a group migration mechanism. Since each group migration requires independent authentication with a new trusted group, the confidence of authentication increases with number of group migrations. The public key and the confidence of its authentication are exposed by the application.

The address space of email addresses is virtually unlimited. It is also vulnerable to the Sybil attack [8] because of the free availability of email addresses. This attack works through the creation of multiple virtual identities so that the honest majority assumption can be violated. This problem is solved by creating trust groups only on the set of parties that have had two way communication. Thus, we are assured that if the end human user has not been attacked in a similar multiple impersonation attack, then the authentication system will be resilient too.

5 Implementation of email authentication plugin

We implemented peer-to-peer sender authentication as a plugin for Thunderbird email client from the Mozilla application suite. The plugin is available for public download at `discolab.rutgers.edu/sam`. This section outlines the design issues, application choices, and practical considerations encountered during its design and implementation. An overview of the plugin architecture is also provided.

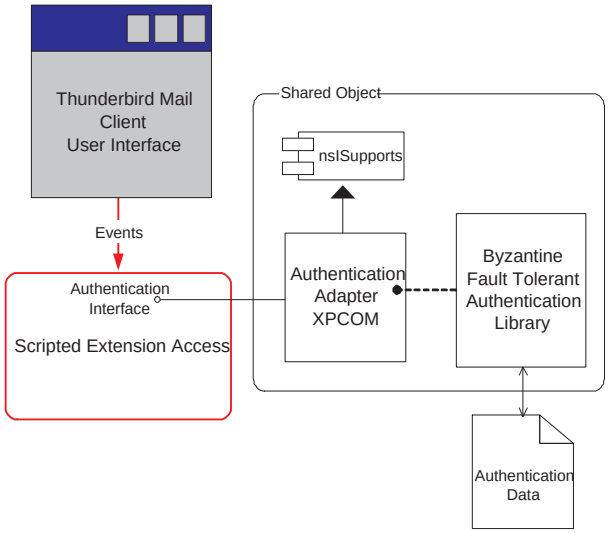

Figure 6: Authentication plugin architecture.

The Mozilla suite of applications [1] allows developers to add new functionality to applications. XPCOM objects are the basic unit of plugin development. These objects allow run time linking and expose their interface through a compiled interface definition file. A compiled XPCOM object can be accessed as a first class Javascript object from the user interface controlling scripts. The user interface itself is defined through the XUL user interface language with Javascript making XPCOM calls on receiving user interface events. The entire package of compiled XPCOM objects, user interface elements, and controlling scripts is referred to as a plugin. Plugins can be installed into any Mozilla application. We followed the standard procedure [3] to embed Byzantine fault tolerant public key authentication in Thunderbird in order to provide peer-to-peer sender authentication.

The architecture of peer-to-peer sender authentication plugin is shown in Figure 6. The Byzantine fault tolerant public key authentication library is statically linked into an XPCOM object that exposes the authentication interface. This interface provides authentication protocol messages to be attached to outgoing electronic mail messages, and consumes authentication protocol messages from incoming electronic mails. The authentication protocol assigns authenticated public keys to known addresses. The authentication interface also contains calls to verify authenticity of public keys associated with email addresses.

6 Experimental evaluation

The objective of experimentation is to characterize client costs, and to establish the suitability of peer-to-peer sender authentication in a real life scenario. The experimentation is done in two stages. The first evaluation is a micro-benchmark consisting of sending and receiving messages from an instrumented byzantine authentication plugin. The second evaluation consists of localized execution of two anonymized email traces, one from a university and another from the industry. The details of the traces are given in Table 2. The university trace is collected from a `sendmail` log behind the spam filter, while the industry trace is collected from the Internet mail gateway ahead of the spam filter. Statistics are collected for data overhead imposed on email messages, cache size at the peers, and the performance of authentication[6].

6.1 Micro benchmarks

A set of micro benchmarks were conducted on a 2.4GHz Intel Pentium 4 desktop running LINUX Fedora Core 5. The objective of micro benchmarks is to determine the latency introduced by the addition of authentication plugin in the email processing path. The added latency of sending and receiving emails was measured for different public key sizes as shown in Figure 7. The sender cost was about 200ms for all the different public key sizes. Sender latency is dominated by message serialization costs and therefore does not depend on public key size. On the other hand, the receiver costs are dominated by the cryptographic operations of digital signature verification and responding to challenges. As shown in figure 7, the receiver costs increase from 85ms at 512 bit keys to about 500ms for 2Kb keys. While both of the costs are within usability limits, one can observe that receiver processing can be done asynchronously in a separate thread. Therefore, one can expect a net addition of about 200ms latency to email operations due to the authentication plugin.

The effect of trusted group size on authentication plugin overhead was also measured as shown in Figure 8. The overhead on the sender increases with increasing trusted group size because of the increasing overhead of serializing larger number of messages for trusted peers. The overhead increased from 160ms at trusted group size of 8 to 220ms for a trusted group of 18 peers. The receiver overhead does not depend strongly on trusted group size and takes about 65ms. The overhead of compression and making function calls across the authentication interface were measured and found to be less than 10ms in all the cases. These overheads and are not sensitive to authentication protocol operational parameters as expected. Sending overhead depends on trusted group size while the

[6]Ideally this data should be collected from real deployment of the authentication plugin but this has practical problems. Firstly, the plugin would need extensive deployment to be a credible source of experimental data. Secondly, privacy fears would prevent many users from allowing collection of detailed usage reports even if these were anonymized.

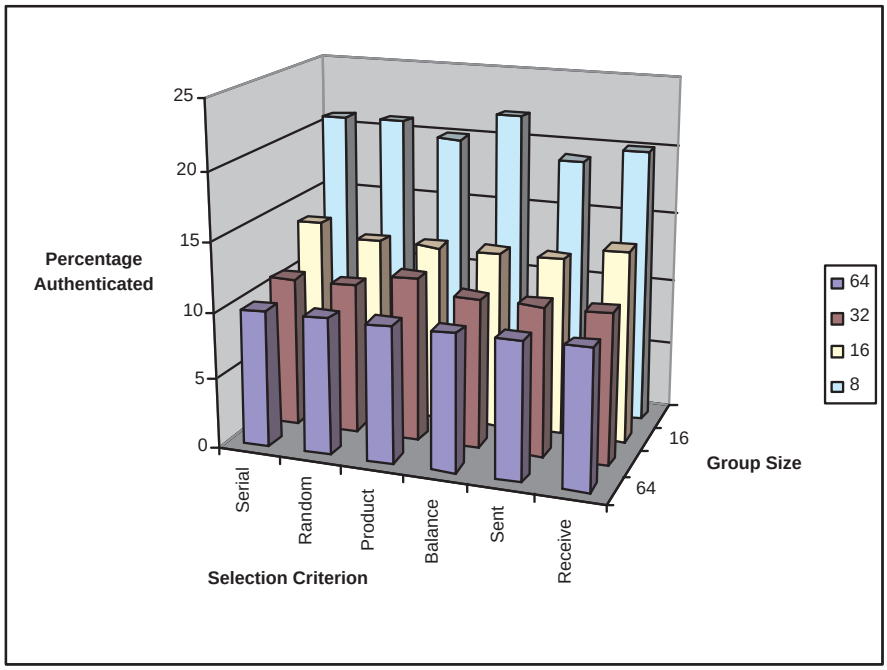

Figure 9: Authentication performance vs. selection criterion and size of bootstrapping group.

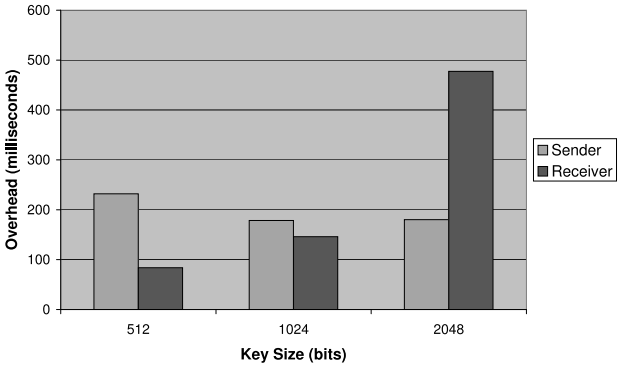

Figure 7: Message processing latency by key length

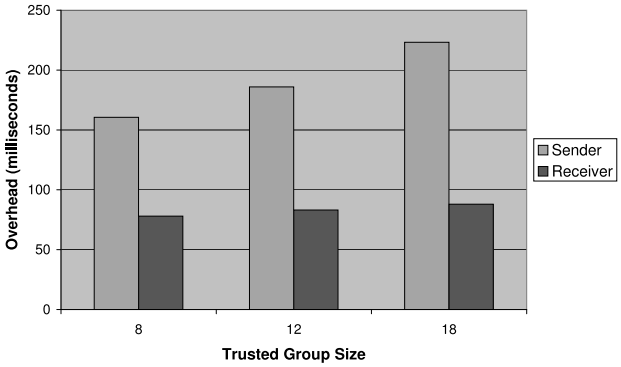

Figure 8: Message processing latency by group size

Trace	Number of messages	Time duration
University	1197043	92 days
Industry	2549767	56 days

Table 2: Email traces used for evaluation.

receiving overhead depends on key size. Since the overhead introduced by the plugin is less than 500ms in all the cases, it is extremely reasonable from a usability standpoint

6.2 University workload

As described in Section 4.1.2, the mail trace is trimmed to email interactions of 53 peers that have bootstrapping groups of size at least 4. A maximum size of 10 is chosen for the bootstrapping group in order to limit the processing time of the trace. The trimmed trace has $873,752$ email messages as compared to the original 1.19 million. This mail trace is used to drive the authentication system with all the peer plugins collocated on a single computer. The results corresponding to message overhead, cache size, and progress of authentication are collected from the logs. We experiment with different values of the following controllable parameters of the authentication system: trusted

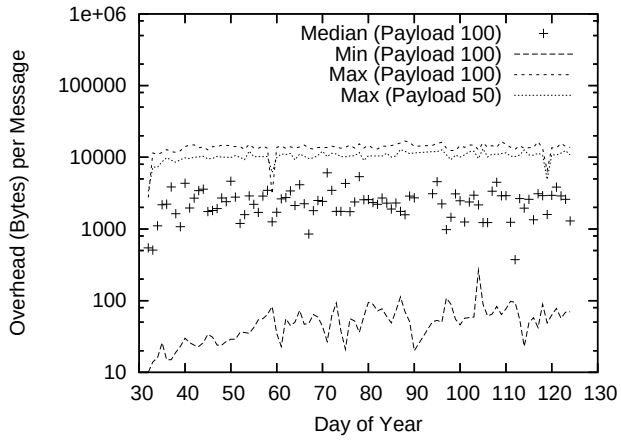

Figure 10: Overhead on email messages

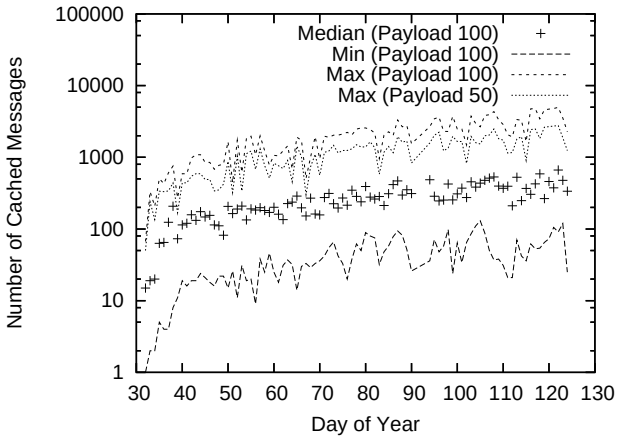

Figure 11: Cached messages with day of year

group size, expiration time for detecting non-liveness of peers, and the maximum number of protocol messages that can be attached to an email message.

6.2.1 Bootstrapping group selection

The authentication protocol performance is sensitive to bootstrapping group selection. In order to ensure progress, the initial candidates are filtered through a two way communication rule as discussed earlier. Experiments were conducted for understanding the suitable bootstrapping group size in email environment. We chose a number of bootstrapping group sizes as shown in Figure 9. A number of selection criteria were developed. The serial and random methods shown in Figure 9 select the bootstrapping peers based by first seen and by uniform random selection on the candidates. The product criterion prefers the peers with higher product of sent and receives messages. The balance criterion prefers the candidates that have balanced bidirectional communication; the absolute difference of sent and received messages is minimized. Sent and receive criteria use the number of sent and received messages respectively.

The performance of authentication is measured by the number of peers that can be authenticated and then averaging over all the mailboxes. As indicated in Section 2, the balanced selection rule has the best completion performance. This is because the underlying protocol requires bidirectional communication for progress. The performance also increases with smaller group sizes because fewer peers can block the progress. Based on these observations, we select the trusted group size to be 10 peers, and use balanced selection criterion for populating bootstrapping groups.

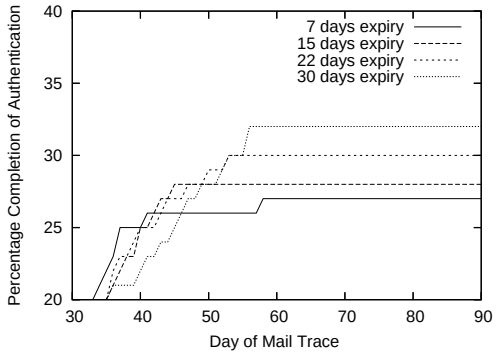

Figure 12: Progrss of authentication vs. message expiry time

6.2.2 Message expiry

The authentication protocol operates as an overlay on the email infrastructure. As a lazy protocol it is susceptible to excessive log growth at the peers. We use an explicit message expiry time and carry it with all the protocol messages. This ensures that each protocol interaction has a finite life time, and thus the log size is bounded[7]. We experimented with a number of practical values for message expiry as shown in Figure 12. The effect of message expiry on authentication performance was found to be marginal. Therefore a moderate message expiry interval of 15 days was used in the experiments.

6.2.3 Message overhead

Authentication protocol messages are piggybacked on email through SMTP extension fields. Because SMTP implementations limit the mail header size, the number of

[7] It was also observed that executing the trace became difficult without having message expiry. Accumulation of stale messages would severely impact the performance.

protocol messages that can be attached to a single email message is limited. In order to understand the overhead introduced by the authentication overlay, we experiment with message payloads of 50, and 100 authentication protocol messages per email message.

The overhead on email messages due to piggybacking of compressed protocol messages is shown in Figure 10. The overhead on messages is bounded on the top by the message payload constraint as shown by the flat maximum observed message overhead in the figure. The median overhead and minimum overhead is shown for payload value of 100. The overhead is positively biased because of a few idle peers. We observe that the payload constraints are reasonable for use with the 32Kb header size limit of SMTP.

6.2.4 Cache usage

Public key infection protocol relies on the lazy propagation of protocol messages. The messages that are not yet delivered need to be cached at participating peers. Using the message payloads of 50 and 100 protocol messages per email, we study the number of cached protocol messages as the authentication protocol progresses. The results are shown in Figure 11.

The number of cached protocol messages shows an increase as the protocol increases. The distribution does not show a significant positive or negative bias as shown by the median being placed in the middle of minimum and maximum values. The initial trend shows an increase in number of cached messages as the protocol authenticates the bootstrapping peers. The median number of cached messages stabilizes as the rate of production and expiry of messages balances out. As shown in the figure, this happens approximately on the 50^{th} day of the trace.

We also note that the maximum permitted payload affects cached messages. As shown in Figure 11, the maximum number of cached messages decreases marginally with decreasing payload. The number of messages is also closely related to the actual size of the cache as shown in Figure 13.

6.2.5 Authentication performance

The authentication protocol results in the authentication of public keys of peers. The progress of authentication is shown in Figure 14. It can be noted that there is a wide disparity between the progress of authentication between the best peer and the average performance of authentication. This gap can be attributed to the fact that most of the email users do not send a lot of messages. The implementation of authentication as an overlay on SMTP limits the rate of progress of authentication. Using a trusted group size limit of 10 peers, payload capacity of 100 messages, and a 15 day message expiry interval, the average peer can authenticate about 35% of its peers of interest in the 92 day run.

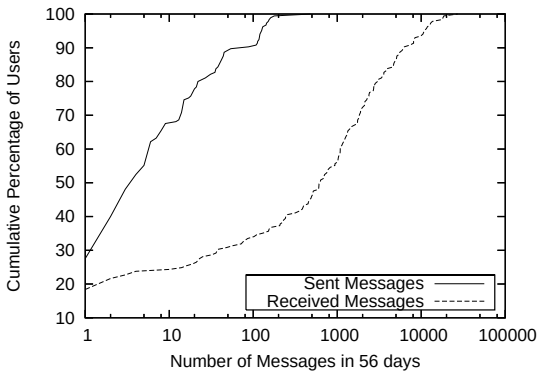

Figure 15: Activity profile of user accounts

It is noteworthy that increasing payloads allow faster completion of the protocol. This is clear from the slower rate of authentication obtained with a payload of 50 messages as compared to 100 messages. This behavior is expected since the progress of the protocol is constrained by the payload limit, which restricts the immediate delivery of all possible messages.

The authentication protocol requires challenge response results from all the trusted peers. However, even one challenge response result from a trusted peer provides some confidence in the authenticity of the public key being authenticated. This "optimistic authentication" is also studied as shown in Figure 14. The average completion of optimistic authentication is at 55% at the end of 92 days, i.e. averaging over all the peers, more than half of the peers have been authenticated. This progress is satisfactory considering that the protocol is backward compatible with the mail infrastructure, has lazy operation, and is fully autonomous.

6.3 Industry workload

The second real trace is collected from the Internet mail gateway of a US corporation. This trace is collected ahead of the spam filter and poses a unique challenge for the authentication mechanism. As shown in Figure 15, while 70% of the addresses received more than 100 messages, less than 50% sent out more than 2 replies over the 56 day period. This is consistent with the large amount of incoming spam and can be contrasted with Figure 4, which shows the distribution on the spam filtered university trace.

The authentication protocol authenticates less than 2% of peers in the industry scenario. This can be contrasted with Figure 14 where the authentication protocol can authenticate a much larger percentage of peers. Analysis

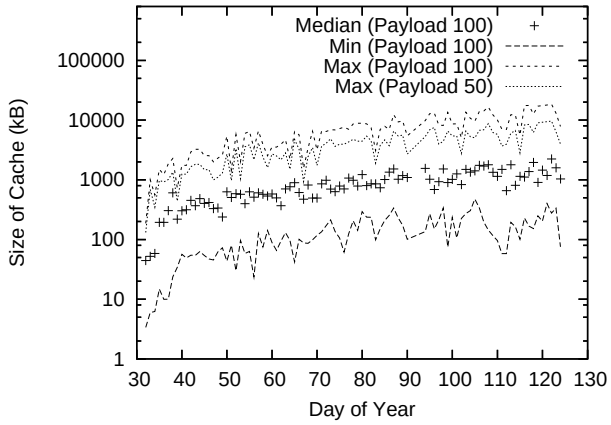

Figure 13: Size of peer caches with day of year

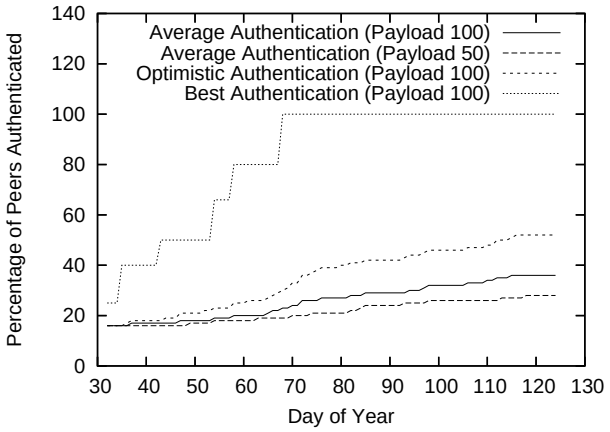

Figure 14: Completion of authentication

of the industry workload shows that senders outnumber the receivers by about 68 to 1. Therefore, most of the senders are not receivers. Since the authentication protocol is required to authenticate the public key of a sender, the few receivers can authenticate only some of the many senders. In order to interpret this result, we considered the instances where a receiver responds to the sender. The industry mail trace had 5 such instances. In two instances, the sender is authenticated by the receiver. We define *effectiveness of authentication* as the fraction of times a receiver can successfully authenticate the sender. We find that the effectiveness of authentication on the industry trace is 40%. In comparison, the university workload has 2301 such instances, and the effectiveness of authentication is 36%. Therefore, the performance of authentication on the industry trace is comparable to that on the university trace.

7 Related work

PGP is a popular public key authentication system. It works with the help of human decisions of trustworthiness that are expressed as public key certificates. PGP uses key rings to store trust relationships that are expressed as public key certifications. It can be noted that the unidirectional trust relationships in PGP can be mapped to a directed trust graph. The vertices of this trust graph are PGP keys and key certificates are the directed edges. Therefore authentication of a node reduces to finding a path to the node in the trust graph. The observed behavior of PGP trust graph has been analyzed in [25, 23, 26]. The trust graph tends to have one very large strongly connected component. For example, biggest key-ring mentioned by `dtype.org` was 1.9GB in April 2002. We can compare the storage requirements by comparing this size with the cache size in our protocol. The storage costs are relatively small, of the order of 10MB, as shown in the experimental evaluation. Therefore, peer-to-peer public key authentication is autonomos and has lighter storage requirements as compared to PGP.

The S/MIME extensions to electronic mail can provide sender authentication and message authentication through the centralized public key infrastructure approach. While this approach is suitable in an organization with a well defined trust hierarchy, it is not suitable for communications that cross organization and trust boundaries. Our solution allows sender authentication across trust boundaries making it suitable for general electronic mail use.

A number of sender domain authentication proposals have been put forward to tackle the spam problem. These include Sender Policy Framework [29], Sender ID [2], and Domain Keys [4]. All of these proposals associate cryptographic material and mail sending policy with the DNS records of domains. This information is used by receivers to detect forged sender addresses. For example, a domain `xyz.com` could nominate a particular server to send all the emails for senders in the domain. The receiving mail transfer agent would check if this policy is being respected, and refuse to accept emails coming from senders in another domain, say `somebody@abc.com`. We believe that there is a fundamental problem with these domain level approaches. The email trust problem is at the mail box level of granularity, whereas the proposed solutions are at the domain level. Using the end-to-end argument [24], only the application that uses sender authentication is best equipped to correctly implement it. For example, users may want to distinguish senders on the same domain and be willing to receive email from `friend@abc.com` but not from `stranger@abc.com`. This kind of fine grained control may be difficult or costly to implement at the domain level. In contrast to these approaches, our peer-to-peer sender authentication mechanism allows sender im-

portance and spam policies to be fine tuned according to the needs of the receiver. An additional benefit of our approach is that the computational cost of cryptographic processing is moved away from the mail gateway to the large number of user desktops.

The widespread use of spam control solutions with false positive errors has reduced the reliability of electronic mail. Garriss et. al. propose the use of social information inherent in the communication patterns to eliminate the false positives of spam filters [10]. However, this work makes stronger assumptions by prohibiting man-in-the-middle attacks and placing complete trust in the attestation servers that manage attestations of social relationship. Walfish et. al. propose another approach to solve the spam problem without introducing false positives [27]. This approach enforces a sending quota in a lightweight fashion but depends on global trust for quota allocators. Unlike these approaches, our sender authentication solution does not require global trust for any entity, resists man in the middle attacks, and provides a useful sender authentication substrate that can be used to prevent false positives of spam filters. There are a number of commercial anti-spam solutions that use a challenge response mechanism to authenticate sender addresses. It can be noted that these solutions affect the usability of email by delaying the delivery of important messages. Our work differs from these solutions in two ways. First, while authentication of a message could be delayed, message delivery is not affected. Second, while the challenge response step of these solutions is vulnerable to the man-in-the-middle attack, our solution resists such attacks.

8 Conclusion and future work

We implement, and evaluate Peer-to-peer Sender Authentication for electronic mail. Our authentication system is automatic, byzantine fault tolerant, and operates without trusted third parties. Authentication is eventually correct under the assumption of honest majority. Our authentication system is incrementally deployable and backward compatible with the existing electronic mail infrastructure Sender authentication is implemented entirely at the mail client in accordance with the end-to-end principle. This enables the creation of user controlled fine grained trust policies that can cross organizational and administrative boundaries. We have implemented sender authentication on the Thunderbird email client as a downloadable plugin at `discolab.rutgers.edu/sam` [18].

Our sender authentication mechanism has been evaluated through micro-benchmarks, and with two real life mail traces. The results show that the overheads are acceptable, and the sender authentication mechanism is effective in real life scanarios. Future work will focus on handling denial of service and collaborative spam control. We plan to develop a peer-to-peer economic incentive scheme to handle denial of service attacks. We plan to create content based spam filters that use collective knowledge from trustworthy peers to improve spam classifiers.

References

[1] Home of the Mozilla Project. http://www.mozilla.org/. This is an electronic document. Date retrieved: February 1, 2007.

[2] Sender ID Home Page. http://www.microsoft.com/senderid. This is an electronic document. Date retrieved: February 1, 2007.

[3] XML User Interface Language. http://www.mozilla.org/projects/xul/. This is an electronic document. Date retrieved: February 1, 2007.

[4] Yahoo! Anti-Spam Resource Center - DomainKeys. http://antispam.yahoo.com/domainkeys. This is an electronic document. Date retrieved: February 1, 2007.

[5] CROCKER, D. H. Standard for the format of ARPA Internet text messages. RFC 822, August 1982.

[6] DEMERS, A., GREENE, D., HAUSER, C., IRISH, W., LARSON, J., SHENKER, S., STURGIS, H., SWINEHART, D., AND TERRY, D. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing* (New York, NY, USA, 1987), ACM Press, pp. 1–12.

[7] DIFFIE, W., AND HELLMAN, M. New Directions in Cryptography. *IEEE Trans. Info. Theory 22* (1976), 644–654.

[8] DOUCEUR, J. The sybil attack. In *Proc. of the IPTPS02 Workshop, Cambridge, MA (USA)* (March 2002).

[9] FIDGE, C. Logical time in distributed computing systems. *Computer 24*, 8 (1991), 28–33.

[10] GARRISS, S., KAMINSKY, M., FREEDMAN, M. J., KARP, B., MAZIÈRES, D., AND YU, H. RE: Reliable email. In *3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (San Jose, CA, May 2006).

[11] KLENSIN, J. Simple Mail Transfer Protocol. RFC 2821, April 2001.

[12] LAMPORT, L. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM 21*, 7 (1978), 558–565.

[13] LAMPORT, L., SHOSTAK, R., AND PEASE, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS) 4*, 3 (1982), 382–401.

[14] LOWE, G. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 1055 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996, pp. 147–166.

[15] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Digital Signature Standard. FIPS PUB 186, May 1994.

[16] NEEDHAM, R., AND SCHROEDER, M. Using encryption for authentication in large networks of computers. *Communications of the ACM 21*, 12 (December 1978), 993–999.

[17] P. RESNICK, E. Internet Message Format. RFC 2822, April 2001.

[18] PATHAK, V. Automatic peer-to-peer mail authentication plugin. http://discolab.rutgers.edu/sam/, 2007.

[19] PATHAK, V., AND IFTODE, L. Byzantine fault tolerant public key authentication in peer-to-peer systems. *Computer Networks 50*, 4 (2006), 579–596.

[20] R. RIVEST, A. S., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (1978), 120–126.

[21] RAMSDELL, B. S/MIME Version 3 Certificate Handling. RFC 2632, June 1999.

[22] RAMSDELL, B. S/MIME Version 3 Message Specification. RFC 2633, June 1999.

[23] REITER, M. K., AND STUBBLEBINE, S. G. Authentication metric analysis and design. *ACM Transactions on Information and System Security 2*, 2 (1999), 138–158.

[24] SALTZER, J. H., REED, D. P., AND CLARK, D. D. End-to-end arguments in system design. *ACM Trans. Comput. Syst. 2*, 4 (1984), 277–288.

[25] SRDJAN ČAPKUN, LEVENTE BUTTYÁN, J.-P. H. Small worlds in security systems: an analysis of the PGP certificate graph. In *Proceedings of the ACM New Security Paradigms Workshop, 2002*. (2002).

[26] STREIB, M. D. dtype.org. http://www.dtype.org/keyanalyze/, 2002.

[27] WALFISH, M., ZAMFIRESCU, J., BALAKRISHNAN, H., KARGER, D., AND SHENKER, S. Distributed Quota Enforcement for Spam Control. In *3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (San Jose, CA, May 2006).

[28] WHITTEN, A., AND TYGAR, J. D. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium* (August 1999).

[29] WONG, M., AND SCHLITT, W. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408, April 2006.

[30] ZIMMERMANN, P. *The Official PGP User's Guide*. MIT Press, Cambridge, Massachusetts, 1995.