
Improving Email Trustworthiness through Social-Group Key Authentication

Vivek Pathak

Department of Computer Science
Rutgers University
Piscataway, NJ 08854

Danfeng Yao

Department of Computer Science
Rutgers University
Piscataway, NJ 08854

Liviu Iftode

Department of Computer Science
Rutgers University
Piscataway, NJ 08854

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. Existing proposals for sender authentication either require infrastructural support or break compatibility with existing email infrastructure. We propose, implement, and evaluate social-group key authentication, an incrementally deployable and backward compatible sender authentication mechanism for email. Our solution requires honest majority instead of trust infrastructure or human input for correctness. In accordance with the end-to-end principle, authentication is implemented at the mail client by executing our previously proposed Byzantine fault tolerant public key authentication protocol [12] as an overlay on top of the mail transport protocol. We evaluated the authentication overhead by instrumenting our Thunderbird authentication plugin with synthetic data and found a user visible latency increase of about 200ms. Real life usability of the authentication mechanism is investigated with two anonymized email traces. Our results show that about 40% of the peers can be authenticated over the 92 day trace period without adding any new messages to the email network. Adding a small fraction of extra email messages permits more than 90% of the peers to be authenticated within a week.

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 a 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 authenti-

cation and message authentication limits the effectiveness and trustworthiness of email. 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 email.

While the original email specification [5, 11] does not address authentication, the S/MIME enhancements [13, 14] 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 or PGP [19]. It allows users to authenticate public keys of other users in a peer-to-peer manner. Its reliance on human judgment of trustworthiness makes it suitable for sophisticated email users [17]. A general purpose email authentication solution must support the following requirements in order to fulfill the needs of decentralized and unsophisticated email users:

1. Operate without depending on centralized third parties for authentication decisions.
2. Provide autonomous operation with minimal human intervention.

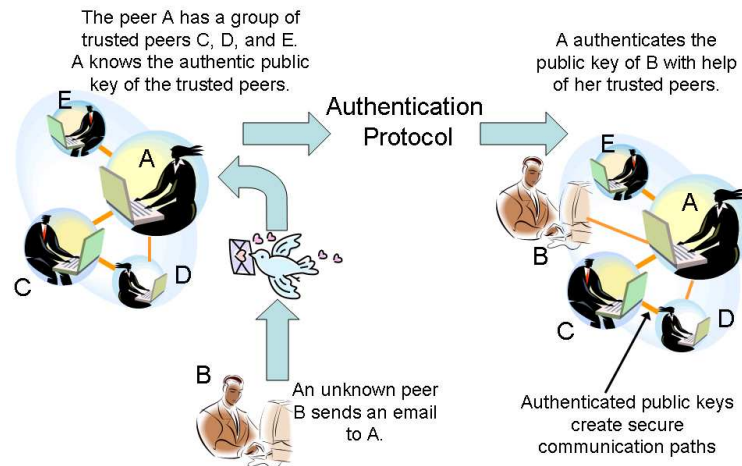


Figure 1: The big picture: public key authentication. A authenticates the public key of B.

1.1 Our solution

Our social-group key authentication proposal for email is described and evaluated in this paper. The proposed solution is an instantiation of our Byzantine fault tolerant public key authentication protocol [12], which supports soft authentication of public keys without centralized infrastructure. The public key authentication protocol runs as an overlay on the email protocol [8], thereby supporting incremental deployment and backward compatibility. Digital signatures [9] generated from the authenticated public keys provide sender and message authenticity to email.

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. Since the underlying public key authentication protocol is autonomous and decentralized, social-group key authentication inherits these characteristics. Authentication is supported in an end-to-end manner without additional infrastructure or human input. Our solution is therefore compatible with the usability requirements described above. It also allows incremental deployment and preserves backward compatibility. In summary, this paper makes the following contributions:

- We implement social-group key authentication for email. Our solution is automatic, Byzantine fault tolerant, eventually correct, incrementally deployable, backward compatible with the existing email 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 email trace, and live experimentation on an instrumented mail authentication prototype.

2 Preliminary

This section provides an outline of our previously proposed Byzantine fault tolerant public key authentication protocol (BPKA) described in [12]. It allows peers to mutually authenticate self-generated public keys. As shown in Figure 1, Alice authenticates Bob’s public key with the help of her trusted peers through a challenge response protocol. Peers whose public keys are already authenticated are called trusted peers. Trusted peers can authenticate public keys and detect malicious behavior under an honest majority assumption.

The BPKA protocol consists of authentication and group management tasks. The operations CHALLENGE RESPONSE, DISTRIBUTED AUTHENTICATION, and BYZANTINE AGREEMENT support autonomous authentication of public keys in the presence of malicious peers. The CHALLENGE RESPONSE operation is used by individual peers to gain evidence of public key authenticity. This evidence is shared among trusted peers through the DISTRIBUTED AUTHENTICATION operation. Lack of consensus on authenticity implies the presence of malicious or faulty peers. The lack of consensus is resolved through the optional BYZANTINE AGREEMENT operation, which permits peers to identify and ignore malicious and faulty peers.

Group management operations maintain honest majority among the trusted peers. The GROUP MIGRATION operation maintains a trusted group of peers by periodically recycling older trusted peers. This protects honest majority in the trusted group by avoiding the accumulation of covertly malicious peers. The BOOTSTRAPPING operation initializes the authentication system by providing the initial trusted group known as the *bootstrapping group*. In this

paper, we use the BPKA protocol as a black box and refer the reader to the original paper for more details about the BPKA protocol [12].

3 Social-group key authentication protocol

The secure association of public keys to email addresses is referred to as *public key authentication* in this paper. This section explains how the previously proposed BPKA protocol [12] is applied to the email environment.

3.1 Email Setup and Security Model

The BPKA protocol assumes that the participating peers are identified by their network addresses, which are email addresses in the context of this paper. Based on this premise, we do not distinguish the email address A from the user who uses that address. We assume that every user U has a public key (K_U) and a private key (K_U^{-1}). Every email message contains the public key of the sender and is signed by the sender using his or her private key.

The BPKA protocol requires that the asynchronous network connecting the peers provide delivery failure notifications for non-existent destinations. The network should support eventual delivery on retransmissions, and not become permanently partitioned. Assuming that temporary failures in the email network are eventually repaired, the email network satisfies these requirements [8].

Public keys are authenticated with help of a group of peers called the trusted group:

DEFINITION 1 (Trusted Group) *The trusted group is used for authenticating public keys of new peers. On authentication of its public key, the new peer becomes part of the trusted group. The public key of every peer belonging to the trusted group is known and trusted.*

The trusted group is initialized from the address book of the user. Note that belonging to another peer’s trusted group does not affect the authentication protocol. Because the authentication protocol requires message transfer between trusted peers, additional extension fields are added to email headers.

DEFINITION 2 (Email Header Extension) *The following email header extension fields are used by social-group key authentication protocol for public key authentication¹:*

- *X-Bft-Auth-PublicKey* : public key of the sender.
- *X-Bft-Auth-Data* : unauthenticated public key of other peers, nonces, cipher text, or trust decisions.
- *X-Bft-Auth-MesgInfo* : the protocol operation that sends out the message and the specific stage within that

¹Bft in the email headers stands for Byzantine fault tolerance.

operation (for operations that have multiple stages). Protocol operations can be one of the following: Email_Peer, Email_Response, or Infer_Trust.

- *X-Bft-Auth-Signature* : digital signature signed with the private key of the sender.

Using SMTP extension header fields for carrying social-group key authentication data provides backward compatibility. The email messages sent by the authentication enabled mail clients would contain social-group key authentication protocol messages, which are processed by the email clients supporting the protocol. The additional protocol messages are ignored by other email clients because email systems should ignore unknown extension headers [11].

3.2 Adversarial Model

We assume the following strong adversarial 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 BPKA 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 [12]. We note that since email servers are widely distributed, a practical value of ϕ is zero for general email communication over the Internet.

Peers in the trusted group can be honest, malicious, or faulty. The protocol does not distinguish between 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 the trusted group has honest majority.

DEFINITION 3 (Honest Majority) *A trusted group has honest majority if fewer than t of the n trusted peers are malicious or faulty, where $t = \frac{1}{3}n$. A peer is malicious if it does not follow the protocol correctly, and faulty if its authentication vote is incorrect.*

For example, a faulty peer may suffer man-in-the-middle attacks causing it to vote incorrectly while a malicious peer may intentionally give wrong authentication votes [12].

3.3 Our Protocol

The purpose of our protocol is to let Alice authenticate Bob’s public key with help from the peers in her trusted group. Our social-group key authentication protocol has the following operations: Email_Init, Email_Peer, Email_Response, and Infer_Trust. The protocol operations are described below along with the exchanged messages.

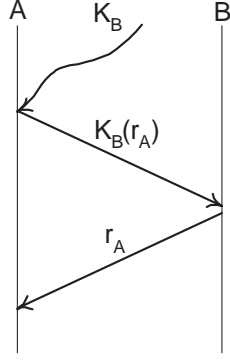


Figure 2: Challenge response in BPKA protocol [12]. A uses the nonce r_A to authenticate the public key K_B of B in absence of man in the middle attack.

For brevity, only the contents of X-Bft-Auth-Data and X-Bft-Auth-MesgInfo email extension headers are described. The remaining extension headers are populated as follows: Public key of the sender is stored in X-Bft-Auth-PublicKey extension header, and the X-Bft-Auth-Signature extension header stores the digital signature created with the sender's private key.

- Email_Init: Alice receives an email message from Bob whose public key K_{Bob} is not authenticated.

$$Bob \rightarrow Alice$$

- Email_Peer: This operation is run by Alice. Alice emails the peers in her trusted group A_1, \dots, A_n for authenticating K_{Bob} , the unauthenticated public key of Bob. The email message has type Email_Peer in the X-Bft-Auth-MesgInfo header, and key K_{Bob} in the X-Bft-Auth-Data header. For all $i \in [1, n]$, we use below formula to represent the email message sent by Alice to peer A_i in her trusted group.

$$Alice \rightarrow A_i \quad K_{Bob}$$

- Email_Response: This operation is run by each A_i with the participation of Bob. As shown in Figure 2, the peer A_i runs the CHALLENGE RESPONSE operation of BPKA protocol [12] and decides if the public key K_{Bob} of Bob is authentic or not. The challenge consists of a random number r_{A_i} chosen by A_i and encrypted with K_{Bob} , the supposed public key of Bob. In response, Bob is expected to recover the random number r_{A_i} chosen by A_i , and demonstrate its ownership of the public key K_{Bob} . The detailed steps of this operation are given below. Each message has the type Email_Response stored in the X-Bft-Auth-MesgInfo header.

$$\begin{aligned} A_i &\rightarrow Bob && K_{Bob}(r_{A_i}) \\ Bob &\rightarrow A_i && r_{A_i} \\ A_i &\rightarrow Alice && T_{A_i}(Bob) \end{aligned}$$

Each peer A_i emails back its *trust vote* $T_{A_i}(Bob)$ to Alice. The trust vote consists of the signed challenge message

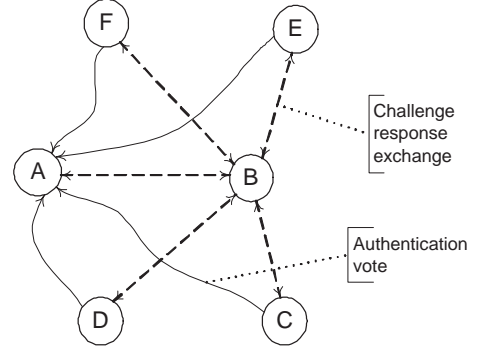


Figure 3: Distributed authentication in BPKA protocol [12]. A authenticates the public key of B by gathering authentication votes from its trusted peers C, D, E , and F .

sent by A_i , the signed response sent by Bob, and a true or false vote on authenticity.

- Infer_Trust: This operation is mainly run by Alice and may also require the participation of Alice's peers and Bob. Alice's inputs are the trust votes received from her peers A_i . If the trust votes are in agreement on the authenticity of K_{Bob} , then Alice decides according to the unanimous decision of her peers. This inference is based on the DISTRIBUTED AUTHENTICATION operation of BPKA protocol shown in Figure 3.

If Alice receives disagreeing trust votes from her peers, she initiates the BYZANTINE AGREEMENT operation of BPKA protocol, which allows Alice to determine who among Bob or her peers is malicious or faulty. Note that Bob needs to participate in the BYZANTINE AGREEMENT step because either Bob or any of Alice's peers may be malicious or faulty [12]. Alice sends the vector of received trust votes to Bob and her peers A_i . On receipt of this message, Alice's peers and Bob exchange the trust vote vectors among themselves. Using the symbol " $|$ " to denote multiple sources or destinations, the messages exchanged in this protocol operation are shown below. Each of the messages contains Infer_Trust in the X-Bft-Auth-MesgInfo header.

$$\begin{aligned} Alice &\rightarrow A_i|Bob && T_{A_i}(Bob) \\ A_i|Bob &\rightarrow A_j && T_{A_k}(Bob) \end{aligned}$$

Alice decides whether or not to trust Bob's key by majority on the trust votes. This part of the authentication protocol also permits Alice and her peers to identify and exclude malicious or faulty peers from trusted groups.

3.4 Security of our protocol

Our social-group key authentication protocol is secure against the adversarial model defined in Section 3.2, assuming the trusted group has honest majority (See Definition 3). Our security is directly based on the security of

the existing BPKA protocol [12]. The proof of security is omitted here.

4 Implementation of email authentication

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 <http://discolab.rutgers.edu/byzantine/>. 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.

The Mozilla suite of applications [3] allows developers to extend application functionality by developing plugins. 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. We followed the standard procedure [2] to embed BPKA library [12] in Thunderbird in order to provide social-group key authentication for email.

The email authentication plugin architecture is shown in Figure 4. *Authentication Adapter* is an XPCOM object which exposes the authentication interface. It is statically linked to the Byzantine fault tolerant public key authentication (BPKA) library [12]. This interface provides authentication protocol messages to be attached to outgoing emails, and consumes the protocol messages from incoming emails. The interface also contains calls to query and authenticate the public keys associated with email addresses. The authentication adapter is used for implementing social-group key authentication. Its functionality is integrated into the Thunderbird email client through the *Scripted Extension Access* module. The authentication plugin is easy to install. It provides automatic email authentication to unsophisticated users.

5 Overlay considerations

We use SMTP extension headers to create an overlay for the social-group key authentication protocol. This maintains compatibility with existing email infrastructure. Running the protocol as an overlay on top of email introduces performance limitations and design constraints. This section investigates these issues in order to choose implementation parameters that are practical in the email environment.

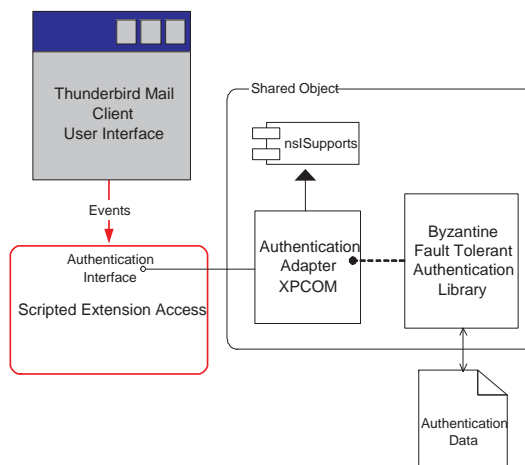


Figure 4: Authentication plugin architecture for the Thunderbird email client.

5.1 Trusted group size limits

The overhead of BPKA protocol was bench-marked through a simulation that investigates the cost of public key authentication [12]. 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 must be selected in order to match the computational and messaging power available in typical email 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. Using a public key size of 1Kb, and ZLIB library for compression, we tested the final header load for different authentication message payloads. A high compression ratio can be achieved on the authentication messages because they are serialized in XML format. Figure 5 shows the size overhead of authentication messages resulting from the university mail trace used in this paper. We choose a payload of at most 300 compressed authentication messages in order to impose less than 10KB overhead on the mail header.

Messaging cost of authentication depends on the trusted group size and the rate of discovery of new peers. The

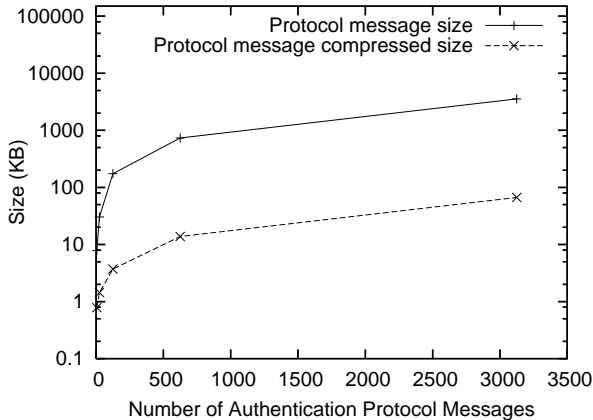


Figure 5: Authentication message size with and without compression. ZLIB compression is used on raw authentication messages created for 1Kb sized public keys.

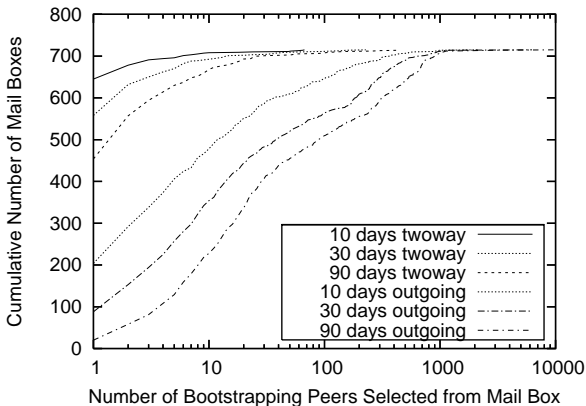


Figure 7: Selection of bootstrapping groups for 715 active user accounts in the email trace.

budget of 300 authentication 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 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 [12] indicate a maximum trusted group size limit of 75 peers for this messaging cost. This upper limit on trusted group size is designed into the system.

5.2 Bootstrapping groups

To determine a meaningful heuristic for generating bootstrapping groups (see Section 2), we analyzed the email

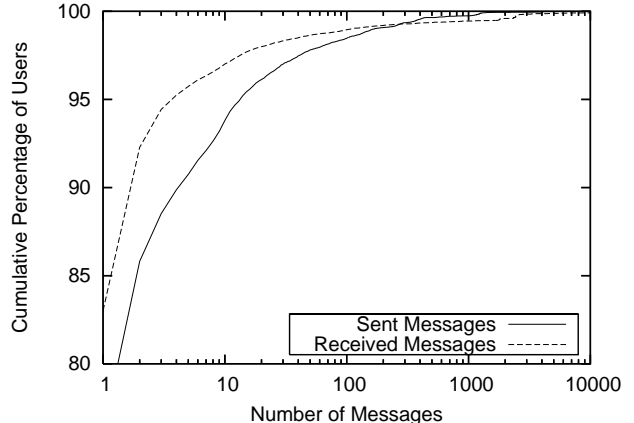


Figure 6: Activity profile of user accounts over the 92 day email trace period. The trace has 27,623 user accounts and 1.19 million emails.

communication patterns available from the anonymized University email trace (described in detail in Section 6). Figure 6 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 cut off 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 against two possible heuristics for generating bootstrapping trust groups: The *Outgoing heuristic* selects bootstrapping peers from destination addresses of outgoing emails. The *Two-way heuristic* selects bootstrapping peers from both the destination addresses of outgoing emails and the source addresses of incoming emails.

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 mailbox 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 7. 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.

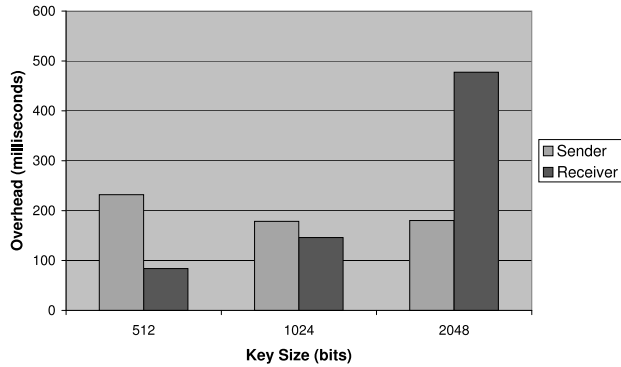


Figure 8: Additional email processing latency for authenticating public keys of different sizes.

5.3 Eager and Lazy modes

The authentication mechanism can be run in lazy or eager modes. In lazy mode, the authentication plugin does not proactively send out any email messages specifically for the key authentication purpose. The protocol messages are therefore transmitted entirely through organically exchanged emails in a piggybacking fashion. In the eager mode, additional plugin generated email messages may be sent out to peers. These messages would be automatically handled and absorbed at the receiving end plugin, and therefore would not change the user experience. We note the downside of eager mode that the added protocol messages may be consumed by spam filters. This problem can possibly be addressed by human means, by asking the mail administrators to disable particular spam filters. However, losing eager mode authentication messages only causes delay because the lazy mode protocol will eventually achieve authentication.

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 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 1. 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. Experiments are also done for comparing the performance of eager and lazy mode authentication.

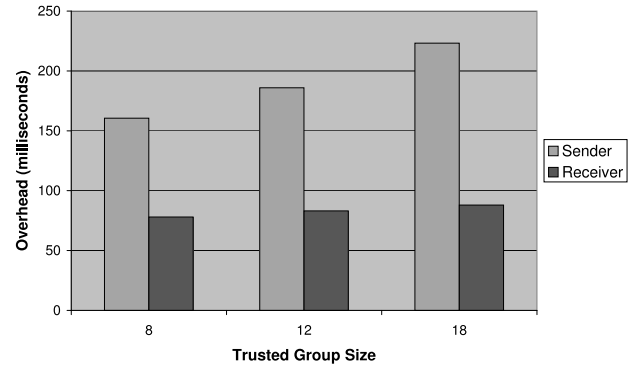


Figure 9: Additional email processing latency for authentication with different trusted group sizes.

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

Table 1: Email traces used for evaluation.

6.1 Micro benchmarks

A set of micro benchmarks was 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 8. 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 8, 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 9. The overhead on the sender increases with increasing trusted group size because of the increasing overhead of serializing a 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 are not sensitive to authentication proto-

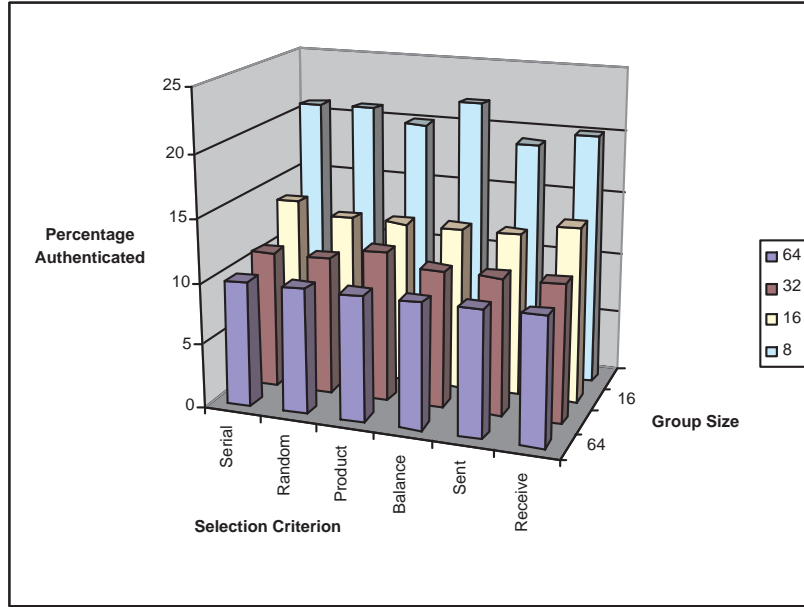


Figure 10: Variation of authentication performance with the size and selection criterion of the bootstrapping group.

col operational parameters, as expected. Sending overhead depends on trusted group size, while the 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 5.2, the mail trace is trimmed to email interactions of 53 peers that have bootstrapping groups of size 4 or more. 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 on a single computer. The resulting message overhead, cache size, and authentication progress are collected from the logs. We experiment with different values of the following controllable parameters of the authentication system: trusted 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 were filtered through a two way communication rule as discussed earlier in Section 5.2. Experiments were conducted for understanding the suitable bootstrapping group size in the email environment. Bootstrapping groups of sizes 8, 16, 32, and 64 were selected as

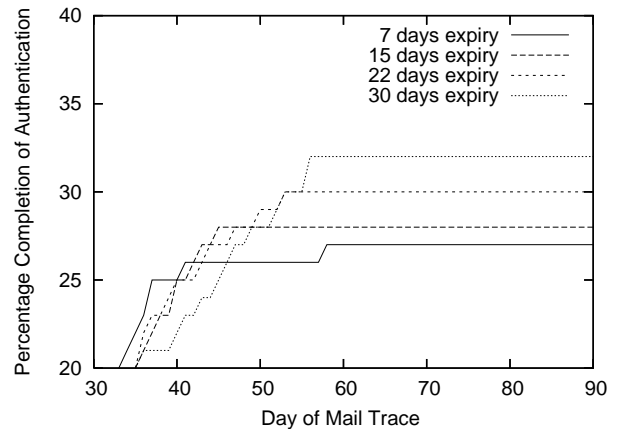


Figure 11: Variation of authentication performance with the expiry time of authentication protocol messages.

shown in Figure 10. A number of selection methods were developed. The serial and random methods shown in Figure 10 select bootstrapping peers by first seen, and by uniform random selection on the candidates respectively. The product criterion prefers peers with a higher product of sent and receives messages. The balance criterion prefers bootstrapping group candidates that have balanced bidirectional communication, i.e. 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. We find that the balanced selec-

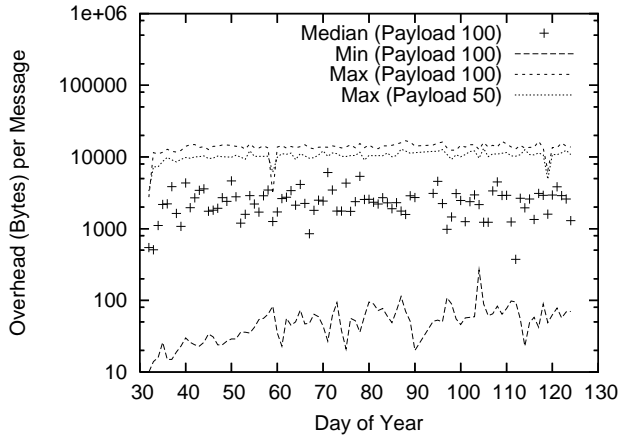


Figure 12: Overhead on email messages carrying the authentication protocol message payload.

tion 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 delay authentication. Based on these observations, we select the trusted group size to be 10 peers, and use balanced selection criterion for populating bootstrapping groups.

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 bounded². We experimented with a number of practical values for message expiry as shown in Figure 11. 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 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 12. Re-

²It was also observed that executing the trace became difficult without having message expiry. Accumulation of stale messages would severely impact the performance.

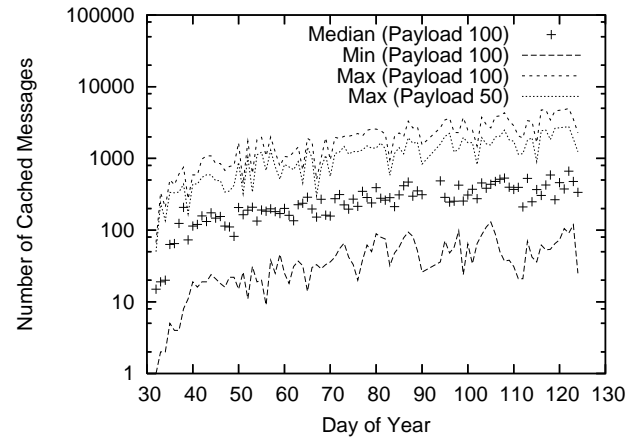


Figure 13: Number of authentication protocol messages cached at email clients.

call the payload constraint of 300 messages and the header size constraint of 10KB applied in Section 5.1. The observed overhead respects the constraint, as shown by the flat maximum message overhead observed for payloads of 50 and 100 messages. The median overhead and minimum overhead are shown for the payload value of 100. The overhead is positively biased because of a few idle peers. We observe that the experimental message payload values of 50 and 100 messages 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 13.

The number of cached protocol messages shows an increase as the protocol progresses. 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 50th day of the trace.

We also note that the maximum permitted payload affects the number of cached messages. As shown in Figure 13, 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 14.

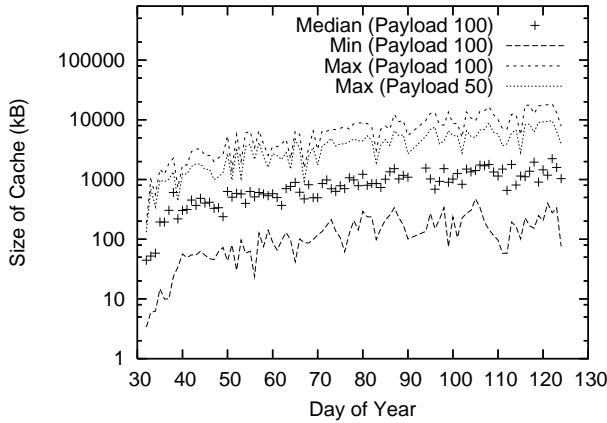


Figure 14: Storage overhead caused by the cached authentication protocol messages.

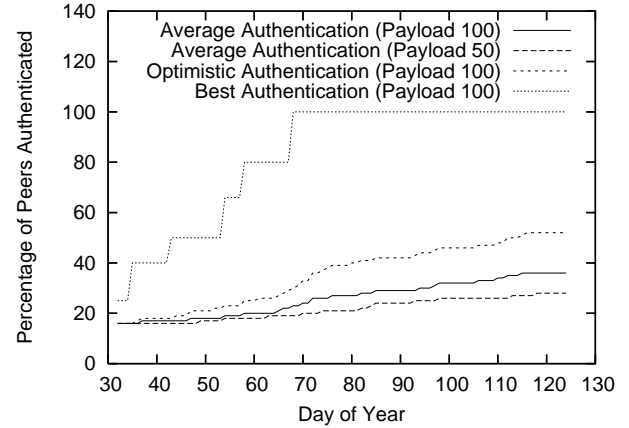


Figure 15: Authentication progress for different authentication message payloads.

Mail Checking Interval	Extra Email Ratio	Time for 80% authentication
Weekly	0.502	14 days
Daily	0.702	2 days
Hourly	7.038	2 hours

Table 2: Overhead and authentication latency for eager mode authentication.

6.2.5 Lazy mode authentication performance

The authentication protocol results in the authentication of public keys of peers. The progress of authentication is shown in Figure 15. 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.

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 15. The average completion of optimistic authentication is at 55% at the end of 92 days,

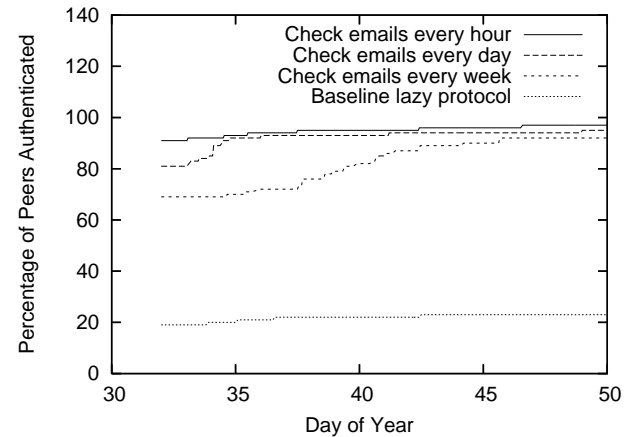


Figure 16: Progress of eager mode authentication with different email checking frequencies.

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.

6.2.6 Eager mode authentication performance

Eager mode authentication performance is evaluated for various eager sending intervals. This assumes that human users typically power up the email client to check for new received emails even if they do not send out any email.

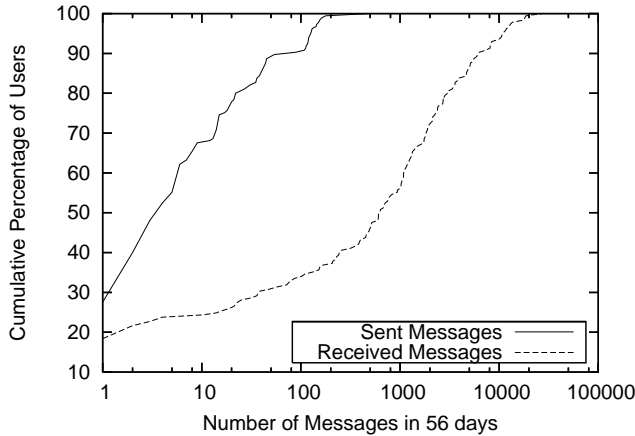


Figure 17: Activity profile of industry user accounts over the 56 day email trace period. The trace has 1.44 million email addresses and 2.5 million emails.

This periodic powering up of the email client is used for sending out the social-group key authentication messages to peers. This speeds up the authentication performance because users who only read emails can also be used for authentication.

We experimented with various periodic intervals for activating eager mode. As shown in Figure 16, the rate of authentication seen by all the peers increases as the periodic interval between eager exchanges is reduced. The eager protocol results in a substantial speedup in authentication performance as compared to lazy baseline authentication. The eager protocol can authenticate more than 90% of the peers within a week if users just check their emails once a day. This is a huge speedup over the slow rate of authentication seen in the lazy case. The overhead introduced by eager mode is measured in terms of ratio of additional email messages sent as compared to the organic email messages captured in the mail trace. The median of ratio overhead introduced by the eager mode is very marginal as shown in Table 2. The time to reach 80% completion is about twice the eager send interval as also shown in Table 2. Thus, the eager mode latency can be used to trade off authentication delay for increased messaging overhead.

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 17, 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 6, 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 15, where the authentication protocol can authenticate a much larger percentage of peers. Analysis 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

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 related approach for public key authentication is the public-space key infrastructure proposed in [10]. This method is applicable to public information like DNS records or BGP routes. Their idea of using multiple peers for observing malicious behavior is similar to our approach.

A number of sender domain authentication proposals have been put forward to tackle the spam problem. These include Sender Policy Framework [18], Sender ID [1], Domain Keys Internet Mail (DKIM) [4], and accredited DKIM (ADKIM) [7]. 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`. These proposed solutions are at the domain level, and are complementary to our user level solution. Our solution aims to achieve individual key authentication, which is at a finer granularity. Using the end-to-end argument [15], 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 complementary to domain level authentication. 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 [6]. 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 [16]. 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 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 social-group key authentication, an automatic, Byzantine fault tolerant authentication system for email. Our authentication system operates without trusted third parties, is incrementally deployable and backward compatible with the existing email infrastructure. It 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 the authentication mechanism on the Thunderbird email client. It is available as a downloadable Mozilla Thunderbird plugin at <http://discolab.rutgers.edu/byzantine/>.

Our authentication mechanism has been evaluated through micro-benchmarks, and with two real life email traces. The evaluation results show that the overheads are acceptable, and the sender authentication mechanism is effective in real life scenarios. We shall be collecting anonymized data from real deployment in order to evaluate the usability of

our solution. Future work will focus on handling denial of service and collaborative spam control. We plan to develop an 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.

Software : <http://discolab.rutgers.edu/byzantine/>

References

- [1] Sender ID Home Page. <http://www.microsoft.com/senderid>, 2007. This is an electronic document. Date retrieved: February 1, 2007.
- [2] XML User Interface Language. <http://www.mozilla.org/projects/xul/>, 2007. This is an electronic document. Date retrieved: February 1, 2007.
- [3] Home of the Mozilla Project. <http://www.mozilla.org/>, 2008. This is an electronic document. Date retrieved: February 15, 2008.
- [4] Yahoo! Anti-Spam Resource Center - DomainKeys. <http://antispam.yahoo.com/domainkeys>, 2008. This is an electronic document. Date retrieved: February 15, 2008.
- [5] D. H. Crocker. Standard for the format of ARPA Internet text messages. RFC 822, August 1982.
- [6] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazières, and H. Yu. RE: Reliable Email. In *3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.
- [7] M. T. Goodrich, R. Tamassia, and D. Yao. Accredited DomainKeys: a service architecture for improved email validation. In *Proceedings of the Conference on Email and Anti-Spam (CEAS '05)*, July 2005.
- [8] J. Klensin. Simple Mail Transfer Protocol. RFC 2821, April 2001.
- [9] National Institute of Standards and Technology. Digital Signature Standard. FIPS PUB 186, May 1994.
- [10] E. Osterweil, D. Massey, B. Tsendjav, B. Zhang, and L. Zhang. Security through publicity. In *HOTSEC'06: Proceedings of the 1st conference on USENIX Workshop on Hot Topics in Security*, Berkeley, CA, USA, 2006. USENIX Association.
- [11] E. P. Resnick. Internet Message Format. RFC 2822, April 2001.
- [12] V. Pathak and L. Iftode. Byzantine fault tolerant public key authentication in peer-to-peer systems. *Computer Networks*, 50(4):579–596, 2006.
- [13] B. Ramsdell. S/MIME Version 3 Certificate Handling. RFC 2632, June 1999.
- [14] B. Ramsdell. S/MIME Version 3 Message Specification. RFC 2633, June 1999.
- [15] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.
- [16] M. Walfish, J. Zamfirescu, H. Balakrishnan, D. Karger, and S. Shenker. Distributed Quota Enforcement for Spam Control. In *3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.
- [17] A. Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium*, August 1999.
- [18] M. Wong and W. Schlitt. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408, April 2006.
- [19] P. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, Massachusetts, 1995.