How to Protect Against a Militant Spammer

Markus Jakobsson* John Linn* Joy Algesheimer

Abstract

We consider how to avoid unsolicited e-mail – so called *spam* – in a stronger adversarial model than has previously been considered. Our primary concern is the proposal of an architecture and of protocols preventing against successful spamming attacks launched by a strong attacker. This attacker is assumed to control the communication media and to be capable of corrupting large numbers of protocol participants. Additionally, the same architecture can be used as a basis to support message integrity and privacy, though this is not a primary goal of our work. This results in a simple and efficient solution that is largely backwards-compatible, and which addresses many of the concerns surrounding e-mail communication.

Keywords: authentication, e-mail, lightweight, spam, strong adversary

1 Introduction

To many people, e-mail has become a crucial tool of daily life, much like the car became an integral part of life for many families some decades ago. E-mail, in fact, to some extent is a substitute for the car, in terms of allowing telecommuting and quick delivery of information. For both of these vehicles of the twentieth century, there is a need for traffic rules and enforcement of the same. Clearly, we would not tolerate a truck parked in the middle of an intersection, the driver handing out flyers. By the same token, there is no reason why the same behavior in the digital domain should be accepted. Until quite recently, though, there has been no suggestion of how to enforce good behavior on the digital roads.

A recent upswing in the amount of *spam*, i.e., e-mail broadcast advertisements, has caused increasing congestion and concern. Spamming is cheap for a spammer but costly to the recipients; inbound spam consumes recipients' time, invades their private environments, and often presents them with offensive content. In response to these concerns, filtering based on source addresses and/or content is becoming increasingly common practice; while this reduces spam, it also renders the overall e-mail system less predictable, useful, and transparent, as some legitimate traffic is rejected. Even with filters in place, spammers continue to be successful in having much of their traffic delivered, by changing source addresses and evading content triggers; to be usefully effective, filters require constant maintenance. More fundamentally, filter-based spam defenses are intrinsically reactive, protecting users against a recurrence of patterns reflecting a previous spam but unable to anticipate the next one that will emerge.

Spammers have many techniques at their disposal: they can purchase bulk lists of e-mail addresses, can harvest addresses from newsgroups and web sites, can use dictionaries to generate addresses, and/or may be able to intercept and extract valid addresses from messages in transit across the Internet. We wish to improve upon prior work (e.g., [10]) by providing countermeasures that are effective against any of these strategies. We propose a strong adversarial model and an efficient solution that meets the adversarial model, making the efforts of even a militant, aggressive spammer largely unproductive. Here, the word "largely" is important: it is necessary to allow for normal e-mail traffic, and to allow two entities that have not previously established a communication link

*{mjakobsson,jlinn}@rsasecurity.com

to communicate with each other (or the utility of e-mail will be severely reduced), so it will still be possible for a spammer to send advertisements in a constrained manner. We suggest, though, that this is not unreasonable, since it shifts the effects of the spammer from an hard-to-trace instigator of traffic jams to an identifiable advertiser that sends directed advertisements, likely in significantly lower quantities.

We consider an attacker that is allowed to eavesdrop on communication lines, and who may collect addresses and other information from valid messages in transit between senders and receivers. He may then use this information, possibly to masquerade as the sender he "stole" the information from. An even more aggressive attacker is allowed the following, ultimate, attack as well: he may, apart from eavesdropping on the communication lines, also perform substitution attacks, viz. substitute a legal e-mail with his advertisement (wherein he may keep selected parts of the old message, such as the sender and receiver information, and any additional elements occurring therein.) In other words, this militant breed of spammer may monitor network traffic, and adaptively insert and substitute messages, hoping to succeed in making a receiver accept a message that was not sent from a party from whom the receiver desires e-mail.

Surprisingly, the above attacks are very easy to defend against, and using only conventional and well analysed cryptographic functions. We demonstrate an efficient and structurally simple solution that protects against an adversary of the above type. Our proposed solution is largely backwards-compatible while interposing new processing modules, and allows for a gradual implementation (in the sense that some entities may choose to employ the scheme, while others do not.) We also consider aspects of integration between the cryptographic methods and common Internet e-mail protocols and processing entities. We consciously shift the computational and storage requirements towards the senders of the e-mails, in order to make the load of the receivers lighter. This serves a dual purpose: First, to discourage spam, and second, to limit the risk of a denial-of-service attack (see e.g., [15] for related issues and methods).

Organization: Following this Introduction, we review related work in section 2. We discuss our model, goals, and supporting building blocks in section 3, and then present our solution in section 4. We summarize conclusions in section 5. In the Appendix, we prove our solution to satisfy our requirements based on the assumptions.

2 Related Work

The conceptually perhaps simplest and least intrusive approach to spamming prevention, and the most commonly used method, is filtering according to senders' addresses and according to message and subject keywords. However, it is not very efficient, as it is easily foiled by adversaries avoiding particular keywords known to be black-listed, and avoiding use of (or camouflaging the use of) certain black-listed sender addresses or domains.

A solution that better protects against spam is to use electronic mail channel identifiers [11]: A receiver assigns a different channel to each sender by giving each of them a unique e-mail address at which to contact him[1]. Incoming mail gets sorted or rejected according to the address sent to and the sender they originate from. Mails originating from unknown senders may be put in a "public" channel. (These public channels are therefore not spam-free.)

The scheme by Gabber et al. [10] applies a related approach. There, a spammer is assumed to obtain e-mail addresses only by compromising servers holding valid addresses

[1]We observe a practical consideration with approaches requiring dynamically created addresses or constructed extensions to existing addresses. This form of integration may not be feasible for all users and e-mail processing environments, particularly those where users lack control over the Message Transfer Agents (MTAs) with which their User Agents (UAs) exchange mail, and where the set of valid and deliverable addresses is fixed by MTA configuration.

(this models both getting e-mail addresses from newsgroups, and buying them from companies compiling e-mail address lists, and similar approaches.) The solution suggested is to use *extended addresses*, which consist of the "normal" address (the so called *core* address) and an extension. The extension is a sequence of characters acting like a password for the e-mail to be accepted, where the extension used by one pair or participants cannot be guessed from a set of other extensions. An extension may only be obtained by sending a request (of a special format that cannot be used for spam) to the desired receiver of an e-mail message; the request may contain a proof of having performed a certain computational task, particular to the pair of sender and receiver (thus implementing a cost for each connection to be set up). Similarly, it may contain a proof of a monetary expense, such as an attached digital coin whose validity can easily be verified and which may be cashed as a "punishment" for spamming.

The weaknesses of both [10] and [11] stem from the fact that the "passwords" are communicated in the clear, allowing an eavesdropping adversary to circumvent the security measures. A difference between [10] and [11] is that Gabber et al. employ computational puzzles to inflict a computational cost on anybody who wants to be given a valid extension.

The idea to require computational payments was first proposed by Dwork and Naor [7]. This was also the first paper to consider how to prevent against spam using cryptographic methods. More recent related work is being pursued in the Penny Black project [17]. In [7], an e-mail has to be accompanied by a correctly evaluated *medium-hard function* (later also called a *puzzle*) in order to be accepted. In contrast, the solution by [10] only requires this setup to be performed once for each pair of communicating participants, which allows the computational cost to be raised without causing communication havoc. (This is an advantage, as by requiring higher computational costs, spamming is further discouraged.)

After the introduction of puzzles in [7], different solutions have been studied by Franklin and Malkhi [9], Gabber et al. [10], and Juels [15], each paper proposing variants with different properties. These papers also use the resulting functions in different contexts, ranging from advertising to prevention of denial-of-service attacks.

The spam problem was also given attention by Cranor and LaMacchia [5], who reviewed the previously mentioned solutions and presented an outline of some regulatory solutions from the legislative realm.

Recently Ioannidis [12] proposed a new filtering tool for spam mails. He introduced a system that encodes specific policies in the email address itself. The policies are chosen by the owner of the email and checked by the user's MTA and/or its MUA. To make sure that the address cannot be altered and the policies remain secret the system uses a MAC and symmetric keys for encryption. Because the policies are encoded in the email address the system does not require local look-up tables to check the policies as they are used in the Tagged Message Delivery Agent System [29]. However, because of the policy extension to the address the generated email addresses become very long and unreadable. This may restrict the set of mailers that can process the mails and yields addresses that are inconvenient for humans to handle. Thus, the system mainly addresses scenarios such as cutting and pasting the email address into web-forms.

All of the above solutions allow an adversary to steal information from messages that allows him later to spam, or to replace portions of messages so that they *become* spam. We consider a stronger model than in previous work, to avoid these problems.

A central building block in our scheme is the message authentication code (MAC for short). A MAC is a keyed one-way function of the input, where the secret key is known by both the generator and the verifier of the MAC, and the validity of the MAC (corresponding to the authenticity of its input) relies on knowing the secret key. We refer to [27] for a review of MACs. An early description of a MAC construction based on a hash function was made by Tsudik in [28]. Preneel and van Oorschot [24] pointed out some problems of this construction, which was later studied and strengthened by Bellare, Canetti and Krawczyk [2]. In the latter paper, a hash-function based MAC design is proposed and proven to be secure based on general cryptographic assumptions.

3 Model, Goals, and Building Blocks

In our model, participants are modelled as senders and receivers, where each participant may act in both of these roles. Our goal is that only those messages originating from *registered* senders will be presented to a recipient user through her primary channel; other messages will be weeded out before presentation. The registration process is triggered automatically when a message arrives from an unregistered sender; the recipient user's attention is not required, and it is unnecessary for the recipient to anticipate and white-list the names of participating senders from whom the recipient desires to receive messages. A party becomes registered by paying an agreed-upon price, which may be either a computational or monetary cost. It must not be possible to substantially reduce the price per working connection. (This allows us to establish lower bounds on the costs of spamming.) Furthermore, a message recipient can retract the registration of any sender at any time, thereby forcing spammers to pay the price of becoming registered for each e-mail message to be sent. The receiver may also implement a categorization of senders, allowing sorting of messages according to meaning, context and priority.

Our suggested solution can, as a side-effect, provide recipients with authentication of message senders. The authentication can be either relative to a pseudonym (for which the identity of the owner is not known), or relative to a certified identity. The former is trivially obtained using our proposed solution, and the latter can easily be obtained if the sender has a public key registered, allowing her to prove knowledge of the corresponding secret key during the setup phase (which only occurs once for each pair of participants.) Note also that the authentication feature does not allow the receiver to convince a third party of the authenticity of a given message (without the implicit cooperation of the sender), i.e., we obtain *private authentication*[2] of messages. (If this property is not desired, then standard digital signature methods can be applied on a per-message basis.) Whereas neither authentication nor private authentication are primary goals of our investigation, they constitute an additional bonus of employing our method. Finally, message privacy (using standard encryption methods) can be obtained based on the primitives constructed for spam protection purposes. Therefore, our solution, while not cryptographically advanced, successfully manages to solve an array of related and functionally interdependent real-world problems, whose exact relations have not previously been clarified.

We envision integration of these methods into a conventional Internet e-mail environment, where messages are submitted using Simple Mail Transfer Protocol (SMTP) [16] and retrieved using Post Office Protocol (POP) [18]. Adaptation to other protocol environments, such as the Internet Message Access Protocol (IMAP) [6] should also be feasible. We anticipate that the cryptographic processing associated with our solution will be performed by proxy processes, interposed on submission and retrieval channels, and therefore that neither User Agents (UAs) nor Message Transfer Agents (MTAs) will require modification in order to incorporate protection against spammers. While participation in the scheme would require deployment of new software (which could be located either on users' PCs or on shared servers), a pragmatic factor which could complicate its adoption, the added facilities would not disrupt users' e-mail processing tools or practices. Overall, the contribution of our work should be seen in terms of the practical issues it addresses, and in the clean and simple solution it provides.

[2] See, e.g., [14] for a description and solution of a similar concept relating to signatures.

3.1 Formal Model

Formally, we denote the sender Sally with S, and her intended receiver Ray with R. In our model, a participant is therefore a computational device, and this can be modelled by a polynomial-time Turing Machine. A *user* is the human or program with access to the machine that corresponds to a participant. For a user to be presented with an e-mail M along with a label describing its sender S, M must first be *accepted* by the receiver R corresponding to this user. In turn, for a receiver R to accept an e-mail M from a sender S, an entry describing the latter needs to be in a list of wanted senders with R. Descriptions of participants get added to this list by paying a setup cost C, but may be removed from the list by R if R (or its corresponding user) so decides. When this occurs, no more messages from S will be accepted by R, and no more messages from S will be displayed to the user (including those already accepted but not already displayed.)

Adversary. An adversary is a party who may corrupt any number of participants. When a participant is corrupted, the entire memory contents and computational resources are made available to the adversary, who also controls what messages are sent by the corrupted party. It is assumed that the adversary can not influence the actions of a participant who is not corrupted. Additionally, the adversary controls the communication channel, and may remove and inject messages at will. He can read all communication between non-corrupted participants. We model the latter by the (stronger) adversarial model in which the adversary may chose a polynomial number of messages and have the MACs of these computed by an oracle before he attempts to perform an attack. (Note, however, that the adversary may not include one of the MACs obtained from the oracle in any of the messages he causes to be sent.) The adversary has the following goals:

1. **Spamming attack**
   One goal of the adversary is to make some k non-corrupted participants $\mathrm{R}_1 \ldots \mathrm{R}_k$ accept each one message $M_1 \ldots M_k$, where these messages are chosen by the adversary, without the adversary having to pay a total cost close to or exceeding $(k-k')C$, where k' is the number of the above messages that are labelled as being sent by a participant who is corrupted.

2. **Authentication attack**
   A second goal of the adversary is to make a receiver R accept an e-mail M and present this to the user as originating from a non-corrupted participant S, when in fact the message did not originate from S.

3. **Sender privacy attack**
   A third goal of the adversary is to convince a non-corrupted participant that a message M was sent by a non-corrupted sender S to some (potentially corrupted) receiver R. (Thus, this is an attack in which the authentication is transferred.)

4. **Message privacy attack**
   The fourth and final goal of the attacker is to determine whether a first message M_1 or a second message M_2 was sent between two non-corrupted participants. More formally, this corresponds to a successful chosen ciphertext attack (see, e.g., [3] for a detailed treatment.)

Defenses against these attacks define basic requirements for our approach. We require that an adversary cannot succeed with the above attacks with more than a negligible probability. More specifically, except with a negligible probability, an adversary cannot (1) succeed with a *spamming attack*; (2) succeed with an *authentication attack*; (3) succeed with a *sender privacy attack*; or (4) succeed with a *message privacy attack*.

3.2 Building Blocks

We apply several elements as building blocks to construct our approach:

Authentication: As an authentication mechanism we use so-called message authentication codes, which we use both to *validate* and *authenticate* the content of transmitted messages. The MAC can be based on a keyed cryptographic hash function h, where the key is known (only) by the sender and the intended receiver(s). We require that it must be a hard problem to determine whether a given string is a valid MAC on a given message m, unless the verifier has access to the secret key. This must hold even after a polynomial number of MACs for chosen messages (other than m) have been seen. The security of the authentication mechanism relies on the assumptions on the underlying hash function. It has been shown that for this to hold, it is necessary and sufficient that it is hard to compute the output of the hash function without knowledge of the secret key, and that the hash function must be (weakly) resistant to collision finding. (We refer to [1, 2] for a thorough treatment.)

In addition to these basic properties, we assume that the hash function h can be modelled by a random oracle [4], in order to use it for key generation purposes. (This assumption may be replaced by another if we use another method for key generation.)

In practice, it is believed that MD5 [25] and SHA-1 [21] have the desired properties required for the hash function.

Symmetric Encryption: To achieve privacy we use a symmetric encryption algorithm, such as the DES [19] or AES [22]. We assume the symmetric encryption to be secure against chosen cipher text attacks.

Public-Key Encryption: For the protocol we use a public-key encryption scheme that is secure against adaptive chosen message attacks [23]. Such a scheme could be RSA [26] or ElGamal encryption [8].

Setup cookies: For each setup, the initiator sends over a *setup cookie* that is evidence of the initiator having paid a cost (or that it is possible to force him to pay this cost.) This may either be implemented using a puzzle, or by some monetary mechanism.

Mail Proxies: For integration purposes, we rely on the ability to interpose proxy processes on inbound and outbound e-mail streams. The outbound case is straightforward: rather than submitting a message directly to an MTA, a sending UA sends it to the proxy, which in turn submits it to the MTA following its processing. In the inbound case, the proxy retrieves e-mail (likely including both spam and legitimate messages) from the MTA, processes them, and stores the messages that qualify for presentation to the user until they are retrieved and deleted by the UA. This case requires more complex processing, as a potentially large number of messages may need to be stored by the proxy, in the form of a shadow maildrop representing a proper subset of the messages retrieved from the MTA. Further, a means should be provided to allow selective access to received messages which are legitimate and desired by a recipient but which lack MACs, whether in the form of an explicit white-list and/or through access to a separate shadow maildrop.

Message Canonicalization Function: When computing and verifying MACs on messages, we first apply a canonicalization function to the message content. Specifically, we exclude whitespace from the input to MAC computation. This step enables the MAC to be successfully validated in the face of certain observed non-transparent characteristics of message transport system components. While this weakens the quality of message integrity provided, as modifications to message whitespace will go undetected, it supports the primary spam protection goal by reducing the chances that "benign" transit processing effects will result in MAC failure at a recipient.

4 Design Principles and Solution

A first design principle is to shift the computational burden from the receiver of an e-mail to the instigator. In order to do this, we want the setup cookie to be verifiable with a minimum of effort by the receiver. In particular, we do not want the receiver to have to perform any decryption for this to be possible, and so, since the setup cookie must be sent in cleartext, it must be specific to the sender (and her public key) and receiver (in order to avoid "cookie kidnappings".) On the other hand, in order to reply to a successful setup request, it is necessary to send the reply in an encrypted fashion. Again, in order to push the computational costs to the sender, we want this action to be efficient for the receiver of the e-mail. We suggest using RSA for this return-channel encryption, as RSA has very low encryption costs. Finally, we want to minimize the amount of storage allocated by the receiver, which is done using the compacting method suggested in [13]. All together, the use of these principles reduce the risks for denial-of-service attacks mounted on potential receivers by flooding these with incorrect setup requests.

A second design principle is for valid extension elements to depend on the message and be such that they cannot be forged by an eavesdropper. This corresponds to not sending the password in the clear, but instead to proving knowledge of the password, where the proof is done with respect to the message sent. This is practically achieved using MACs, carried within new message header elements.

We now present the specifics of our solution:

1. **Request for Setup:**
   If Sally sends an e-mail to Ray that carries no MAC extension header element, her computer gets an automatic response stating that a setup needs to be performed. Her e-mail is not delivered, but bounces back. Along with the returning message, a homepage or ftp address may be sent, from which appropriate software may be downloaded. This download should be needed no more than once per sender; once the downloaded software is active on the user's computer, it will be able to perform the setup protocol with arbitrary numbers of recipients, operating as a proxy interposed on the user's submitted and retrieved mail streams. Alternately, server-based proxy processes could be interposed on behalf of the user, obviating the need for installation of software on the user's computer; while such executables could be shared on behalf of multiple users, they would maintain state data on a per-user basis.

2. **Setup:**
   Sally's proxy process generates and sends Ray a setup cookie which includes a public key y_S to which only Sally's proxy knows the secret key. Note that successful execution of the setup protocol requires that Sally provide a valid reply address and process messages received at that address, an additional factor that may operate as a deterrent to would-be spammers. (If y_S is a *certified* public key, this allows Ray to create a link to the identity of the sender. Note, however, that it does not need to be certified for the scheme to work.)

   Ray's proxy verifies the correctness of the cookie (corresponding to verifying that the puzzle solution or the piece of digital cash is valid), and selects a symmetric key K_{RS} uniformly at random from the set of possible keys. (We explain later how this can be performed in a way that conserves resources for Ray.) Ray's proxy then encrypts K_{RS} using y_S, and sends the resulting ciphertext[3] to Sally (using a header facility to distinguish a setup message). On receiving this message, Sally's proxy computes K_{RS}. Her proxy then stores (Ray, K_{RS}) in a list of all such access keys. All future e-mails from Sally to Ray will be processed using this key.

[3]We note that an attacker could replace this ciphertext, in which case the subsequent transmission of the e-mail from Sally to Ray would be rejected. To avoid this attack, one could require Ray to authenticate the message. We do not do this, in order to reduce Ray's computational effort.

3. **Sending a message:**
Let m be the message Sally wants to send to Ray, $\mu = h(m)$, and let $e = MAC_{K_{RS}}(\mu)$. This value e is carried within an extension header element when the message m is sent to Ray.

4. **Receiving a message:**
Ray's proxy looks up (or computes) K_{RS}, given the alleged sender of the e-mail, Sally. Ray's proxy calculates e as above, and accepts the e-mail iff the same result as in the extension header element is obtained. An accepted e-mail gets delivered to Ray, just like a normal e-mail would normally be delivered after having been received. Otherwise, the e-mail is considered as a request for setup (see above) and the message goes undelivered.

Remark 1: Compact key management. The symmetric key can be generated by Ray as $K_{RS} = h(K_R, S, i_S)$, where h is a hash function, K_R is Ray's secret key, S is Sally's name and i_S is a sequence number that is increased for each setup request by Sally. In our model, this amounts to the same as choosing this key uniformly at random from the same set of keys. However, it allows Ray a very compact representation of the key; he merely has to store his secret key and a small counter indicating the value of i_S.

Remark 2: Message Privacy. For the message encryption we propose to use a symmetric algorithm in the CBC mode [20]. The secret key can be generated by Sally as $K_S = h(K_{RS}, count)$[4], where h is a hash function, K_{RS} is the shared key from the MAC, and *count* is a counter increased for each message sent between Sally and Ray. (We note that this key may be generated without any extra communication.)

Remark 3: Categorization. In the above, we have only described filtering to avoid unwanted e-mail. It is trivial to use the same methods to categorize the incoming e-mails according to sender-specific categories. It is possible for a sender to request a higher (or lower) priority by indicating this in some field that gets authenticated along with the message. This is relatively safe from abuse since if a user is considered to have abused the increased priority setting he may lose his registration.

Remark 4: Mailing List Support. For a solution to satisfy the functional needs of Internet e-mail, it must support mailing lists that are expanded by intermediaries. Senders to such lists generally cannot identify all of a list's subscribers, a property that is often desirable for privacy reasons. Therefore, it seems inappropriate for a list sender to perform a registration operation with each subscriber. Instead, we recommend that the list expander register with its subscribers, and that a list sender register with the list expander. Given the leveraged fan-out of list-based distribution, it may be appropriate for list expanders to require more costly or frequent registrations than many individual users would specify.

This basic protocol satisfies requirements 1-3 as presented in section 3, and requirement 4 as well for the protocol in which encryption is used to obtain message privacy. We prove these claims in the Appendix.

5 Conclusions

Today, spam disrupts the productive use of e-mail communications. Incurring only minimal costs, a spammer can invade the privacy of untold numbers of recipients, consuming their time and resources. Spammers are adopting ever more insidious and aggressive practices, but most recipients remain either unprotected or screened by reactive filters that require continuous maintenance and often drop legitimate messages. Overall, the situation is untenable, and appears almost certain to worsen if left untreated.

[4]It is also possible to use the key K_{RS}, but to achieve stronger security and easier analysis, we propose to use different symmetric keys for the two applications.

In this paper, we have proposed a technical approach to level the field. Through our suggested methods, a recipient gains control, becoming able to adjust the effort that a sender must expend in order for a message to be shown to the recipient. We have presented a general adversarial model, and have built on existing cryptographic methods and prior work to construct a corresponding solution. We have considered integration strategies to incorporate these methods into common Internet e-mail protocols and processing environments. We hope that these results can contribute to effective practice, helping e-mail to serve its users rather than its would-be abusers.

References

[1] M. Bellare, R. Canetti, H. Krawczyk, “Message authentication using Hash Functions-The HMAC Construction,” RSA Laboratories’ CryptoBytes,1996.

[2] M. Bellare, R. Canetti, H. Krawczyk, “Keying Hash Functions for Message Authentication,” Advances in Cryptology-Proceedings of Crypto’96.

[3] M. Bellare, A. Desai, E.Jokipii, P. Rogaway, “A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation,” FOCS, 1997.

[4] M. Bellare, P. Rogaway, “Random oracles are practical: A paradigm for designing protocols,” First Annual Conference on Computer and Communications Security, ACM, 1993.

[5] L.F. Cranor, B.A. La Macchia, “Spam!,” Communications of ACM 98. Available at `http://www.research.att.com/~lorrie/pubs/spam`.

[6] M. Crispin, “Internet Message Access Protocol - Version 4rev1,” Internet RFC-2060, December 1996.

[7] C. Dwork, M. Naor, “Pricing via Processing or Combating Junk Mail,” Advances in Cryptology-Proceedings of Crypto’92, pp.139-147.

[8] T. ElGamal, “A Public-Key Crytosystem and a Signature Scheme Based on the Discrete Logarithm,” Advances in Cryptology - Proceedings of Crypto ’84, pp. 10-18.

[9] M. Franklin , D. Malkhi,“ Auditable Metering with Lightweight Security,” Financial Cryptography ’97, pp.151-160.

[10] E. Gabber, M. Jakobsson, Y. Matias, A. Mayer, “Curbing Junk E-Mail via Secure Classification,” Financial Cryptography ’98.

[11] R.J. Hall, “Channels: Avoiding Unwanted Electronical Mail,” Communications of ACM ’98. Available at `ftp://ftp.research.att.com/dist/hall/papers/agents/channels-long.ps`

[12] J. Ioannidis, “Fighting Spam by Encapsulating Policy in Email Addresses,” Proceedings, Internet Society 10th Network and Distributed System Security Symposium, San Diego, February 2003.

[13] M. Jakobsson, “Mini-Cash: A Minimalistic approach to E-Commerce,” PKC ’99.

[14] M. Jakobsson, K. Sako, R. Impagliazzo, “Designated Verifier Proofs and Their Applications,” Advances in Cryptology-Proceedings of Eurocrypt’96.

[15] A. Juels, J. Brainard, “Client Puzzles : A Cryptographic Countermeasure Against Connection Depletion Attacks,” Proceedings of the 1999 Network and Distributed System Security Symposium, Internet Society, pp.151-165.

[16] J. Klensin, ed., “Simple Mail Transfer Protocol,” Internet RFC-2821, April 2001.

[17] Penny Black Project, http://research.microsoft.com/ research/sv/PennyBlack/.

[18] J. Myers, M. Rose, “Post Office Protocol - Version 3,” Internet RFC-1939, May 1996.

[19] NBS FIPS Pub 46-1, "Data Encryption Standard," U.S. Department of Commerce, 1988.

[20] NBS FIPS Pub 81, "DES modes of operation," U.S. Department of Commerce, 1980.

[21] NBS FIPS Pub 180-1, "Secure Hash Standard," U.S. Department of Commerce, 1995.

[22] NIST FIPS Pub 197, "Advanced Encryption Standard," U.S. National Institute of Standards and Technology, 2001.

[23] M.Naor, M. Yung, "Public-key cryptosystems provably secure against chosen chipertext attacks," Proceedings of the 22nd Annual Symposium on Theory of Computing, ACM, 1990.

[24] B. Preneel, P. van Oorschot, "On the security of two MAC algorithms," Advances in Cryptology-Proceedings of Eurocrypt'96,

[25] R. Rivest, "The MD5 Message-Digest Algorithm," RFC1321, 1992.

[26] R.Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, v.21, n.2, 1978, pp. 120-126.

[27] B. Schneier, *Applied Cryptography.*

[28] G. Tsudik, "Message authentication with one-way hash functions," ACM Computer Communications Review,v.22,n.5,1992, pp.29-38.

[29] Tagged Message Delivery Agent project, http://tmda.net/.

A Proofs

Theorem 1: Our system protects against the *spamming attack* in the chosen message attack model. This means that we first allow an adversary E to receive valid MACs on a polynomial number of messages that E chooses. Then, if an adversary is successful in making some k non-corrupted participants $\mathrm{R}_1 \ldots \mathrm{R}_k$ accept each one message $M_1 \ldots M_k$, he must first pay at least a cost $(k-k')(C-\epsilon)$, where k' is the number of the above messages that are labelled as being sent by a corrupted participant, and ϵ is a small constant that corresponds to the maximum savings possible by batching several setup computations.

Proof of Theorem 1: *(Sketch)*
Let us assume that the receiver R accepts an e-mail with the belief that it was sent by a sender S. Since the receiver is assumed not to be corrupted, he only accepts e-mails with valid MACs w.r.t. the believed sender. The adversary E cannot gain any information from corrupting participants not involved as sender or receiver in any of the message transfers, since the secret keys used for the MACs are chosen independently at random. For the same reason, he cannot compute more than some t keys after t accepted setup sessions. (If this does not hold then the assumption that the hash function used to generate the shared secret keys is a random oracle does not hold.) Moreover E can not get any information about the secret key during its transmission, or he can break the encryption scheme used for sending K_{RS}. Therefore, the adversary must either (1) corrupt somebody who has paid the setup cost C, or (2) pay the setup cost $C - \epsilon$ for the e-mail to be accepted, or (3) has to produce the valid MAC without the key given in the setup phase. (This is the case since no such key or portion of it will be given out unless the expected setup cost C has been paid.) In the former case, S is one of the k' corrupted players, and in the second S is one of the k "paying" participants. We argue that the third case only can occur with a negligible probability. This is so since the MAC according to the assumptions is resistant against chosen message attacks. Thus, the total cost paid by the adversary is $(k - k')(C - \epsilon)$.

Theorem 2: Our system protects against the *authentication attack*, i.e., the adversary is not able to make a receiver R accept an e-mail M and present this to the user as originating from a non-corrupted participant S, when in fact the message did not originate from S.

This follows from the proof of Theorem 1, and the fact that non-corrupted participants will only accept e-mails with valid MACs corresponding to the apparent sender.

Theorem 3: Our system protects against the *sender privacy attack*, i.e., an adversary is not able to convince a non-corrupted participant E that a message M was sent by a non-corrupted sender S to some (potentially corrupted) receiver R.

Proof of Theorem 3: *(Sketch)*
Due to the property that the MACs are assumed secure against a chosen message attack, it is not possible for a third party to determine that a given message was sent by one of R and S (without attempting to determine which one) without the involvement by either of these two parties. Let us therefore assume that one of these two parties collaborates with the third party to try to convince the latter that the other participant sent a certain message. If M was sent from S to R than it has a valid extension computed with the correct key K_{RS}. However, this key is also known to R, and thus, it is not possible to determine whether R or S originated the message. Therefore, if one of the participants were to try to convince a third party of who the sender is, the third party will not trust the proof, as the transcript could have been computed by either R or S.

Theorem 4: If the encryption mode of our scheme is employed, then our system protects against the *message privacy attack*, i.e., the attacker is not able to determine whether a first message M_1 or a second message M_2 was sent between two non-corrupted participants.

This follows directly from the symmetric cipher being assumed to be secure against chosen message attacks.