SELS: A Secure E-mail List Service

Himanshu Khurana, Adam Slagell, Rafael Bonilla
NCSA, University of Illinois
Urbana-Champaign, USA
{hkhurana, slagell, bonilla}@ncsa.uiuc.edu

ABSTRACT

Exchange of private information content among a large number of users via *E-mail List Services* is becoming increasingly common. In this paper we address security requirements in that setting and develop a new protocol, SELS (a Secure E-mail List Service) that provides confidentiality, integrity, and authentication for e-mails exchanged via lists. In addition, SELS also protects against the use of lists for e-mail spamming. We have developed a prototype of SELS in Java, and integrated it with the Eudora e-mail client.

Categories and Subject Descriptors

H.4.3 [**Communications Applications**]: Electronic Mail; C.2.0 [**General**]: Security

Keywords

Electronic mail, Mailing List, Security

1. INTRODUCTION

Electronic mail (or *e-mail*) has become one of the most widely used means of daily communication. As more and more user communities are engaging in collaborative tasks, use of *e-mail list services* is also becoming common; i.e., e-mails exchanged with the help of a list server (examples of commonly used list server software include LISTSERV [24] and majordomo [27]). The increasing popularity of e-mail list services (ELSs) for exchanging both public and private information content can be gauged from the fact that there are over 300,000 registered LISTSERV lists while only 20% of those serve public content [11].

Considerable work has been done in providing various security solutions that enable secure exchange of e-mail between two parties; e.g., solutions for message confidentiality, integrity, and authentication [23, 9, 34, 10, 14]. However, little or no work has been done towards providing similar security solutions for ELSs. In this paper we argue that on one hand many of the solutions for two-party e-mail exchange are not applicable to ELSs and, on the other hand, solutions that may not be suitable for the former are applicable to ELSs. We present SELS, a Secure E-mail List Service, that provides solutions for confidentiality, integrity, authentication, and anti-spamming for ELSs without using any additional components and imposing minimal overhead on existing protocols. We have implemented our scheme in Java and integrated it with the Eudora e-mail client; thus illustrating that the solution works with existing e-mail systems.

The rest of this paper is organized as follows. Section 2 discusses related work and the contributions of this paper. Section 3 discusses SELS entities and presents the SELS encryption scheme. Section 4 presents the SELS protocol, which is analyzed in Sections 5 and 6. Section 7 discusses implementation and scalability, and Section 8 concludes.

SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA

2. RELATED WORK AND CONTRIBUTIONS

Confidentiality. Solutions for confidentiality of e-mails exchanged between two parties include PEM [23], PGP [34], mediated-RSA [9], identity-based encryption [10], and identity-based mediated-RSA [14]. Extending these solutions to solve the confidentiality problem of ELSs would not work because the extensions would all require the sender to encrypt the e-mail to the list server who would then decrypt and re-encrypt the e-mail to the list subscribers. This would provide the list sever with access to e-mail plaintext and, consequently, the list server would become an attractive attack target and a potential single point of security failure for access to e-mails exchanged and archived on all mailing lists managed by the server. Cryptographic hardware solutions where the decryption and re-encryption would be done in hardware could reduce the vulnerability, but these solutions are expensive and have been broken in the past [12]. Instead, we develop a software-based proxy encryption scheme that enables the list server to transform encrypted e-mails between senders and receivers without requiring access to the plaintext. Proxy encryption functions such as those developed in [7, 21, 28] enable the transformation of ciphertext encrypted with one key into ciphertext encrypted with another key without the transforming agent having any knowledge of the corresponding private keys. We develop a new proxy encryption scheme that is suitable for ensuring confidentiality in SELS and, as opposed to these previous works, generates the proxy key using a distributed key generation protocol. One could potentially view the confi-

dentiality problem as a key distribution (or key agreement) problem where the list subscribers would be given (generate) a symmetric key for encrypting e-mails, and the key would change whenever users subscribe to or unsubscribe from the list; i.e., similar to the key distribution problem for secure group communication. However, neither key distribution solutions (e.g., the scheme of Wong *et al.* [33]) nor key agreement solutions (e.g., the scheme of Kim *et al.* [22]) apply to the problem at hand because they would both require the list subscribers to be online and execute operations for updating the symmetric key whenever users subscribe to or unsubscribe from the list; a notion that goes against commonly accepted ways of e-mail use.

Integrity and authentication. The use of digital signatures (e.g., RSA or DSS signatures) is a commonly accepted solution for verifying the integrity and authenticity of e-mails [23, 34]. However, their widespread use is hindered by the underlying requirement of large-scale certificate distribution and revocation. Fortunately, this problem can be easily solved in the ELS setting as entities that manage the lists (e.g., the list server and the list moderator) can distribute certificates to list subscribers and validate their status (with respect to list membership).

Anti-spamming. Spam has become a huge problem and, consequently, many solutions have been proposed; e.g., solutions based on filters [4], proof of computation [15, 16], policy enforcement [20], ticket servers [2], and others [13, 17]. These solutions can be applied to both two-party e-mail exchange and ELSs but high success rates and widespread use have yet to be demonstrated. In ELSs one could attempt to keep list subscribers' e-mail addresses private (at the list server) and use them for filtering spam. However, users often advertise their e-mail addresses and associations with organizations (e.g., on their website), which imply corresponding list memberships and provide spammers with access to the list. In addition, the privacy of list subscriber e-mail addresses is vulnerable to implementation errors [32]. A solution based on the use of digital signatures [31] that would not work for two-party e-mail exchange would work for ELS. It would not work in the former case because as pointed out in [25], the solution would require large scale certificate distribution and revocation, and would block the use of e-mail as a common form of initiating relationships and exchanging credentials. It would work for ELSs because, as stated above, the problem of certificate distribution and revocation is easily solved in the ELS setting. Furthermore, we show that in ELSs the use of keyed hash-MACs (HMACs) can also provide a solution; one that is cheaper than using digital signatures.

Contributions. In this paper we present solutions for confidentiality, integrity, authentication, and anti-spamming for e-mails exchanged via ELS. Our confidentiality solution is based on a new proxy encryption scheme and allows archiving of e-mails in encrypted form. We provide integrity and authentication with digital signatures, and anti-spamming with HMACs. A related area that we do not focus on in this paper is development of certified e-mail protocols that provide delivery guarantees; e.g., [1, 8].

3. AN INTRODUCTION TO SELS

In this section we discuss the entities involved in the SELS protocol and present the encryption scheme used by SELS to achieve confidentiality. We assume that all entities have access to an e-mail client and an e-mail server through which they send and receive e-mails.

3.1 SELS Entities

The following entities are involved in the SELS protocol:

• *List Moderator (LM).* LM is a user (or process) that creates a list to be maintained at the list server, authenticates users, and helps them subscribe to and unsubscribe from the list. To create a list, LM establishes key material with the list server, and to subscribe a user, it distributes key material to the joining user. LM signs and distributes certificates of list members, provides certificate validation with respect to list membership, and helps recover from compromise of a user or the list server. We assume that LM is an autonomous entity (in particular, it is not controlled by the list server).

• *List Server (LS).* LS creates lists, maintains membership information (e-mail addresses and key material), adds and removes subscribers based on information received from LM, and forwards e-mails sent by a valid list subscriber to all current subscribers of that list. Also, LS optionally archives e-mails exchanged on the list in encrypted form.

• *Users/Subscribers.* Users subscribe to lists by sending join requests to LM, obtaining key material from LM, and then sending key material to LS to complete the subscription process. Users unsubscribe by sending leave requests to LM, who then requests LS to delete the leaving member's information.

3.2 SELS Encryption Scheme

In this section we present the SELS public-key encryption scheme $\mathcal{E}$, which is based on the discrete log problem like El Gamal. $\mathcal{E}$ specifies an encryption transformation function that enables LS to transform an e-mail message encrypted with the sender's public-key into messages encrypted with the receivers' public keys. We first present a notation for El Gamal and then describe $\mathcal{E}$.

Let $\mathcal{E}_{eg} = (Gen, Enc, Dec)$ be the notation for standard El Gamal encryption [18]. Gen is the key generating function. Hence $Gen(1^k)$ outputs parameters (g, p, q, a, g^a) where g, p and q are group parameters, (p being k bits), a is the private key, and $y = g^a \bmod p$ is the public key. The Enc algorithm is the standard El Gamal encryption algorithm and is defined as $e = (mg^{ar} \bmod p,\ g^r \bmod p)$, where r is chosen at random from Z_q. To denote the action of encrypting message m with public key y, we write $Enc_{PK_y}(m)$. Dec is the standard El Gamal decryption algorithm and requires dividing mg^{ar} (obtained from e) by $(g^r)^a \bmod p$. We assume all arithmetic to be *modulo* p unless stated otherwise.

We denote the SELS asymmetric encryption scheme by $\mathcal{E} = (IGen,\ UGen, AEnc, ADec, \Gamma)$. Here $IGen$ is a distributed protocol executed by LM and LS to generate group parameters g, p and q, private keys K_{LM} and K_{LS} and public keys $PK_{LM} = g^{K_{LM}}$ and $PK_{LS} = g^{K_{LS}}$. K_{LM} is simply a random number in Z_q chosen by LM, and K_{LS} is a random number chosen by LS. $UGen$ is a distributed protocol executed by user U_i, LM, and LS to generate private keys for U_i and LS. $UGen(K_{LM}, K_{LS})$ outputs private keys K_{U_i}, K'_{U_i} and the public keys $PK_{U_i} = g^{K_{U_i}}$, $PK'_{U_i} = g^{K'_{U_i}}$. K'_{U_i} is called user U_i's *corresponding private key* and is held by LS. Furthermore, it is guaranteed that $K_{U_i} + K'_{U_i} = K_{LM} + K_{LS} \bmod q$. This protocol requires U_i, LM, and LS to generate random numbers and add/subtract

them from K_{LM} and K_{LS}. *AEnc* and *ADec* are identical to *Enc* and *Dec* defined above. $\Gamma_{K'_{U_i}}$ is a transformation function that uses user U_i's corresponding private key to transform messages encrypted with U_i's public key into messages encrypted with the *list key* $K_{LK} = K_{LM} + K_{LS}$ mod q; though no entity knows the value of K_{LK}. It takes as input an encrypted message of the form $(g^{rK_{U_i}}M, g^r)$ and outputs $(g^{rK_{LK}}M, g^r) = (g^{rK'_{U_i}}g^{rK_{U_i}}M, g^r)$. Once *UGen* has been executed for users U_i and U_j, then sending a message between the users requires user U_i calling $AEnc_{PK_{U_i}}$, *LS* calling $\Gamma_{K'_{U_i}}$ followed by $ADec_{K'_{U_j}}$, and user U_j calling $ADec_{K_{U_j}}$. The encryption scheme $\mathcal{E}$ is correct because $ADec_{K_{U_j}}(ADec_{K'_{U_j}}(\Gamma_{K'_{U_i}}(AEnc_{PK_{U_i}}(m)))) = m$.

The encryption scheme $\mathcal{E}$ is secure if it retains the same level of security as the standard El Gamal scheme against all adversaries $\mathcal{A}$, and if *LS* cannot distinguish between encryptions of two messages even with access to multiple corresponding private keys. The proofs are given in Appendix A.

Theorem 1 Let $\mathcal{E} = (IGen, UGen, AEnc, ADec, \Gamma)$ be the SELS encryption scheme. $\mathcal{E}$ is CPA (chosen-plaintext attack) secure against the *List Server* and any adversary $\mathcal{A}$.

4. SELS PROTOCOL

We now present the SELS protocol. As illustrated in Figure 1, the protocol specifies steps for creating a list, subscribing users, sending e-mails, and unsubscribing users. We assume that all entities, namely, the users, *LM* and *LS* have (or can obtain and trust) each other's public-key certificates for encryption of user subscription e-mails and for signature verification of all e-mails (e.g., PGP certificates or those from an external PKI). *LM* plays a crucial role here in that it distributes certificates of list subscribers for signature verification; i.e., it replies to certificate requests by sending a signed list of all (or a subset thereof as requested) currently subscribed list members' certificates. *LM* also supports certificate revocation with respect to list membership by sending a list of certificates belonging to members who have unsubscribed with monthly subscription updates (a more immediate form of revocation is enforced by the unsubscribe protocol). *LM* does not, however, provide validation of the private keys in the certificates as a Certificate Authority would — only validation with respect to list membership. (For large lists *LM* may be unable to provide certificate distribution but it would still provide certificate validation and subscriber would obtain each other's certificates for signature verification from an external PKI). We distinguish SELS keys from external PKI keys by placing a *bar* on top of the PGP/external PKI keys. $Enc_{\overline{PK_i}}$(m) denotes the encryption of message m with public-key $\overline{PK_i}$, and $Sig_{\overline{K_i}}$(m) denotes the message m along with its signature using private key $\overline{K_i}$.

4.1 Creating a List

To create a new list L, *LM* and *LS* execute the following steps:

1. *LM* begins the execution of *IGen* and generates parameters $(g, p, q, K_{LM}, g^{K_{LM}})$, and associates the key pair (K_{LM}, PK_{LM}) with the list.
2. *LM* then sends *LS* a message with the values g, p, and q, and the new list ID L. Formally, *LM* $\longrightarrow$ *LS*: $Sig_{\overline{K_{LM}}}$("Create" List L, g, p, q).
3. *LS* then completes the execution of *IGen* by choosing a new private key K_{LS}, computing public key PK_{LS} = $g^{K_{LS}}$ and associating the key pair with the list.

Both *LM* and *LS* implicitly agree that the sum K_{LK} = K_{LM} + K_{LS} (mod q) is the *list key* but neither knows its value since neither knows the other's private key. The list is now ready for subscription.

4.2 Subscribing Users

To subscribe user U_i to list L, U_i, *LM* and *LS* execute the following steps:

1. U_i sends a signed "join" request to *LM*. Formally, $U_i \longrightarrow LM : Sig_{\overline{K_{U_i}}}$("Join" List L).
2. *LM* authenticates U_i and begins the execution of *UGen* by generating a random value r, a temporary key TK_{U_i} = $K_{LM} + r$ mod q for U_i, and a ticket encrypted with *LS*'s public-key containing the value r.
3. *LM* then sends the values g, p, and q, the temporary key, and the ticket to U_i. Formally, *LM* $\longrightarrow$ U_i: $Enc_{\overline{PK_{U_i}}}$ ($Sig_{\overline{K_{LM}}}$(TK_{U_i}, Ticket, g, p, q)) where TK_{U_i} = $K_{LM} + r$ mod q and Ticket = $Enc_{\overline{PK_{LS}}}$ ($Sig_{\overline{K_{LM}}}$(L, U_i, r)).
4. On receiving this message from *LM*, U_i generates a random value r' and computes his private key K_{U_i} = $TK_{U_i} + r'$ mod q. U_i also computes a symmetric key H_{U_i} to be the hash (message digest) of r'; i.e., H_{U_i} = $h(r')$. This key will be used for computing HMACs [6] of e-mails exchanged between U_i and *LS* for message authentication, integrity, and anti-spamming.
5. U_i then sends the value r' to *LS* encrypted with $\overline{PK_{LS}}$ along with the ticket received from *LM*. Formally, $U_i \longrightarrow LS$: $Enc_{\overline{PK_{LS}}}(Sig_{\overline{K_{U_i}}}$("Join" L, r')), Ticket.
6. *LS* authenticates the ticket via *LM*'s signature, obtains r and r' from this message, and computes the corresponding private key K'_{U_i} = K_{LS} - r - r' mod q. *LS* also computes key $H_{U_i} = h(r')$.

This completes the execution of *UGen*. Note that the sum of keys K_{U_i} and K'_{U_i} is also K_{LK} (the list key) and that neither *LM* nor *LS* knows the user's private key K_{U_i}. *LM* also adds U_i's certificate to the list of currently subscribed users and makes this certificate available on request.

4.3 Sending E-mails

To send an e-mail to the list L, sender U_i, *LS*, and all receivers U_b ($b \neq i$) execute the following steps:

1. U_i first signs the e-mail m with his private key $\overline{K_{U_i}}$[1] and then encrypts it with his public key PK_{U_i} using the SELS encryption function[2] *AEnc* (let X =

[1]To prevent numerous attacks, it is essential that a signature be applied to a message prior to encrypting it [5].

[2]In our implementation, we employ a hybrid form of encryption for greater efficiency where the bulk e-mail content is encrypted with a symmetric key cipher and the symmetric key is then encrypted using SELS.

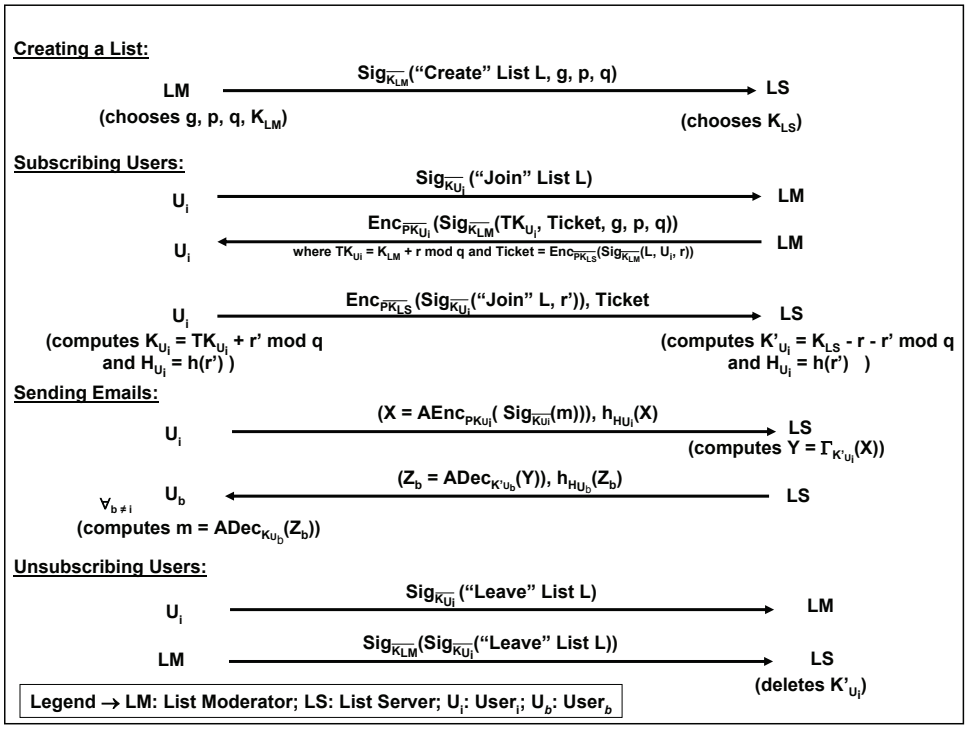

Figure 1: The SELS Protocol

$AEnc_{PK_{U_i}}(Sig_{\overline{K_{U_i}}}(m)))$. U_i also computes the HMAC of X using key $H_{U_i} = h_{H_{U_i}}(X)$.

2. U_i then sends to LS (1) the encrypted e-mail message, and (2) HMAC of X. Formally, $U_i \longrightarrow LS : (X = AEnc_{PK_{U_i}}(Sig_{\overline{K_{U_i}}}(m))), h_{H_{U_i}}(X)$.

3. LS verifies the HMAC with key H_{U_i} and then transforms the message by computing $Y = \Gamma_{K'_{U_i}}(X)$ using U_i's corresponding private key K'_{U_i}.

4. To forward the e-mail to every user U_b who is subscribed to list L, LS computes and sends to U_b (1) a decryption of Y with the private key K'_{U_b} using algorithm $ADec$ — we call this value Z_b, and (2) a HMAC of Z_b using key H_{U_b}. Formally, $LS \longrightarrow U_b$: $(Z_b = ADec_{K'_{U_b}}(Y)), h_{H_{U_b}}(Z_b)$.

5. Receiver U_b verifies the HMAC with key H_{U_b} and decrypts the e-mail from Z_b using his private key K_{U_b} with algorithm $ADec$. The receiver can then verify the digital signature on the decrypted e-mail.

LS can also forward e-mails to LM by decrypting them with K_{LS}; LM can then read the e-mails by decrypting with K_{LM}. Similarly, LM can also send a secure e-mail to the list by encrypting it with PK_{LM} and sending it to LS.

4.4 Unsubscribing Users

To unsubscribe from list L, user U_i, LM, and LS execute the following steps:

1. U_i sends a signed unsubscribe request to LM. Formally, $U_i \longrightarrow LM$: $Sig_{\overline{K_{U_i}}}$("Leave" L).

2. LM verifies the signature, co-signs the request, and sends it to LS. Formally, $LM \longrightarrow LS$: $Sig_{\overline{K_{LM}}}($ $Sig_{\overline{K_{U_i}}}$("Leave" L)).

3. LS verifies LM's signature and deletes U_i's e-mail address and key K'_{U_i}. After unsubscribing, any e-mail sent to list L will not be forwarded to U_i.

LM can also force an involuntary departure of U_i (e.g., in case of compromise) by sending a signed message to LS. Note that the SELS unsubscribe operation does not affect other subscribers; i.e., it does not suffer from the "1 affects n" scalability problem typically found in group key management protocols [29].

4.5 Providing Access to Archived E-Mails

Existing (insecure) ELSs often provide subscribers access to archived e-mails. To provide access to archived e-mails in SELS, LS first archives e-mails in encrypted form. Specifically, e-mails are archived after executing the Γ(X) function on e-mails received from users; i.e., all archived e-mails are encrypted with the list key K_{LK}. Second, in order to give a particular list subscriber access to archived e-mails, LS decrypts the e-mails with the subscriber's corresponding private key and sends it to the subscriber. The subscriber then decrypts the e-mails with her private key.

In analyzing the information leaked to an adversary that compromises SELS entities, we note that if current list sub-

scribers are granted access to all archived e-mails since the inception of the list, then the adversary only has to compromise a single list subscriber to get access to all of those e-mails. Therefore, we limit the adversary's access by using *key epochs* whereby the list key K_{LK} is changed every epoch, and e-mails are not archived beyond the current epoch. (If required, non-sensitive e-mails such as those describing the list's mission can always be archived in clear-text for the life of the list.) To change K_{LK}, LM chooses a random number r and sends it to the list subscribers via LS. On receiving this message, each subscriber U_i adds r to its private key $K_{U_i} \pmod q$. At the same time, LS chooses a random value r' and adds it to all the corresponding keys $\pmod q$. The new list key is the old value added to r and $r' \pmod q$.

5. PROTOCOL ANALYSIS

In this section we informally analyze how the SELS protocol satisfies our goals of confidentiality, integrity, authentication, and anti-spamming. Constructing formal correctness arguments based on a complete and formal protocol specification is beyond the scope of this paper. In this section we focus on adversaries that can observe, insert, and modify e-mail messages and address those that can compromise SELS entities in Section 6.

The integrity and authentication goals of SELS to ensure that an adversary is not able to masquerade as a list subscriber or implement undetected modifications are trivially satisfied via digital signatures. Furthermore, we address the certificate distribution problem by having the list moderator distribute list member certificates and provide certificate validation with respect to list membership.

SELS provides confidentiality of e-mail messages against passive adversaries by the satisfaction of the following two security properties:

- *E-Mail secrecy*: it is computationally infeasible for any passive adversary outside of current list membership (including the list server) to decrypt e-mails.
- *Forward e-mail secrecy*: a passive adversary who was a former list member cannot decrypt e-mails after he has unsubscribed from the list.

E-Mail secrecy is supported by the SELS encryption scheme, which ensures that a passive adversary (or the list server) must be able to break standard El Gamal encryption in order to read e-mails sent with SELS (as stated in Theorem 1). Forward e-mail secrecy is ensured by the unsubscribe protocol where LS deletes a user's corresponding private key when that user unsubscribes from the list. Consequently LS does not decrypt and forward any subsequent e-mails for that user and, therefore, the user cannot read the e-mails. We do not discuss backward secrecy because such a goal is contradictory to providing access to archived e-mails.

Anti-spamming is provided by the use of digital signatures and HMACs. All receivers associate certificates with list subscribers of a given list alias and reject any e-mail received from the list whose signature cannot be verified with the sender's certificate. However, digital signature verification can be expensive and HMACs as used in Figure 1 provide an alternate, inexpensive means of preventing spamming. Furthermore, by using HMACs LS actively participates in spam prevention, which it would not be able to do with digital signatures since the signatures are encrypted when the e-mail is received by LS for forwarding. With HMACs LS uses key H_{U_i} to cryptographically verify that an e-mail was sent by user U_i and receiver U_b uses key H_{U_b} to verify that the forwarded e-mail was sent by LS. Using these keys LS never forwards an e-mail to the list that is not sent by a valid subscriber and receivers do not accept e-mails that are not sent by LS; i.e., no user outside of a list can ever send an e-mail to the list.

6. HANDLING LIST SERVER COMPROMISE

Over a period of time it is possible that the list server gets compromised. In fact, this possibility was the primary motivation behind the use of our proxy encryption scheme. In this section we analyze the harm that an adversary who has compromised LS can do, and provide recovery mechanisms that eliminate the adversary's advantage.

When an adversary compromises LS he gets access to all corresponding private keys and HMAC keys. Using these keys the adversary cannot violate the confidentiality, integrity, and authenticity of e-mails because the corresponding private keys cannot be used to decrypt any e-mails and all e-mails are signed. The adversary can spam lists using the HMAC keys but with limited impact since the digital signature verification will fail. We assume that the compromise of LS is eventually detected; e.g., if the adversary uses LS to spam lists then that would indicate a potentially compromised LS and Allen [3] provides numerous practical mechanisms to detect server compromises. We provide an efficient recovery mechanism in Figure 2, which is executed by LM and eliminates the adversary's advantage in knowing these keys. The recovery mechanism requires LM to send an e-mail to the (re-instated) LS and an e-mail to the list subscribers via LS.

It is possible that while the adversary has compromised LS, he also simultaneously compromises a list subscriber, say U_i. The adversary now has access to U_i's keys K_{U_i} and $\overline{K_{U_i}}$ in addition to all corresponding private keys. This would allow him to compute K_{LK} for the subscriber's list and, consequently, the private keys of all list subscribers and the list moderator. Using these keys the adversary would be able to send e-mails as U_i but would not be able to violate the integrity and authenticity of e-mails sent by other subscribers as they would be signed. The adversary would be able to violate the confidentiality of e-mails in that he would be able to read e-mails sent in the current epoch; however, he would be able to do that even if he comprises U_i alone. Having access to LM's private key K_{LM} in combination with control over LS's functions would allow the adversary to add subscribers to the list; however, e-mails from these subscribers would not be accepted by other subscribers because their certificates would not be signed by LM. We assume that this simultaneous compromise is eventually detected and provide a recovery mechanism that eliminates the advantage of an adversary who compromises LS and user U_i in Figure 2. The recovery mechanism cannot use the keys available to the adversary and, therefore, requires LM to send an email to the (re-instated) LS and individual e-mails to every (uncompromised) list subscriber. This is still significantly cheaper than requiring new subscriptions. We assume that a simultaneous compromise of LS and LM is unlikely, but if it does happen then users must request

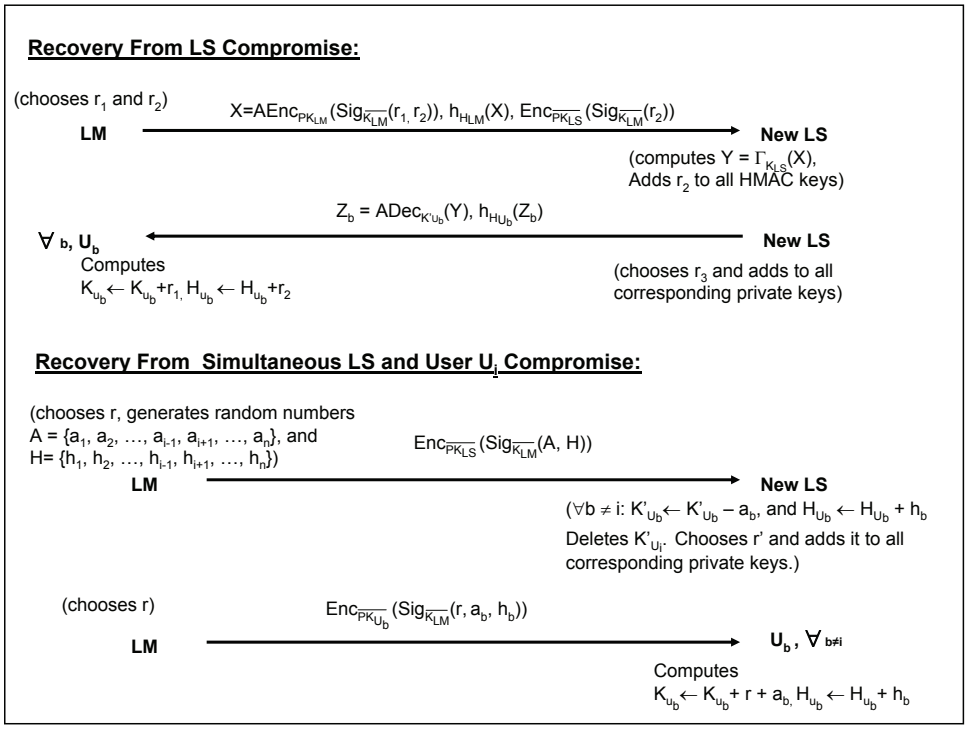

Figure 2: Recovery Mechanisms

for a new subscription to that list after LM and LS are re-instated.

7. IMPLEMENTATION AND SCALABILITY

We have developed a prototype of the SELS protocol in Java and integrated it with the Eudora e-mail client using Eudora's command-line interface and filters on the Windows platform. We use the GnuPG toolkit [19] for standard public-key encryption and signature operations. We describe the implementation in terms of the following four components: (1) Eudora Interface, (2) Crypto Utilities, (3) List Management, and (4) E-mail Processing.

The *Eudora Interface* and *Crypto Utilities* components are common to all entities. The Eudora interface allows the *List Management* and *E-Mail Processing* components to send e-mail messages via a Eudora client connected to a standard e-mail server using Eudora's command-line interface. The crypto utilities component includes:

•*Encryption and decryption routines.* In practice it would be inefficient to use SELS for encrypting e-mails. Instead we use a hybrid form of encryption/decryption where the e-mail is encrypted/decrypted with a random symmetric key using the triple-DES symmetric encryption/decryption algorithm, and the symmetric key is encrypted/decrypted with 1024 bit $AEnc$, $ADec$, Γ algorithms (using a 160 bit subgroup). The routines have been developed using the Java *BigInteger* class and the Java *crypto* library.

•*base64 encoding/decoding routines.* These routines are required to convert binary data into ASCII text (which can then be sent across standard e-mails systems) and have been implemented using the OsterMiller library [30].

•*Random number generation.* This has been implemented using the Java *secureRandom* class.

•*SHA-1 HMAC generation and verification.* This has been implemented using the Java *Mac* class. For e-mails sent by users to LS, the HMAC is computed over the concatenation of the triple-DES encrypted e-mails and the encrypted triple-DES key (i.e., encrypted using $AEnc$). For e-mails forwarded by LS to receivers, the HMAC is computed over the concatenation of the triple-DES encrypted e-mails and the encrypted triple-DES key (i.e., after being decrypted with $ADec$ but not yet in clear-text).

•*Interface to GPG toolkit.* Allows encryption, decryption, signature generation, and signature verification of e-mails using any entity's PGP keys.

The *List Management* component provides functions for creating lists, subscribing and unsubscribing users, and generating and managing SELS keys for users and lists. The component differs for each entity depending on its role in the SELS protocol. For the list moderator this component provides routines for (1) generating a key pair K_{LM} and PK_{LM}, (2) associating it with the list address L, (3) sending a message (via e-mail) to the list server to create list L, (4) subscribing users by authenticating them and generating temporary keys and tickets for them, (5) unsubscribing users by co-signing requests and sending a message to LS, and (6) replying to user certificate requests by sending a

signed list of all (or a subset thereof as requested) currently subscribed users' certificates. For the subscriber this component provides routines for (1) sending a "join" message to LM to join list L, (2) generating and managing a public/private key pair based on a temporary key received from LM, (3) sending a "join" message to LS with the ticket received from LM, and (4) sending an "unsubscribe" message to LM. For the list server this component provides routines for (1) creating a list L based on the message received from LM by generating a key pair K_{LS} and PK_{LS}, and associating it with the list address, (2) subscribing users by verifying the ticket included in the subscribe request from the users and generating and managing corresponding private keys for them, and (3) unsubscribing users by deleting their corresponding private keys based on a message received from LM.

The *E-Mail Processing* component provides higher-level routines that use the *Crypto Utilities* component for encrypting, decrypting, signing, and verifying e-mails using the SELS and GnuPG algorithms. The component accepts a clear-text e-mail message (created with any text editor) processes it and sends it to the desired entity via the Eudora interface component. The component differs slightly for each entity; for the list moderator and subscribers it provides processing for sending and receiving e-mails, while for the list server it provides processing for forwarding e-mails to all list members.

The bulk of the computational overhead of the SELS protocol is imposed on the list server. In particular, the expensive operation is decryption of e-mails forwarded to a list with every list subscriber's corresponding private key. We implemented this operation in C using the GNU multiprecision (gmp) library and measured the time taken for the operation to be 0.83 ms on a 1.8 GHz dual-processor Mac G5 desktop machine (p being 1024 bits and q being 160 bits). That means, using hybrid encryption of e-mails on such a desktop machine a list server would be able to handle lists with hundreds of members without any noticeable delays (and potentially support thousands of members with high-end server machines).

8. CONCLUSION

In this paper we provide solutions for confidentiality, integrity, authentication, and anti-spamming for e-mails exchanged via e-mail list services. We have developed a prototype of SELS in Java and integrated it with the Eudora e-mail client. In the future we will develop a plug-in for Eudora and integrate SELS list server key management and e-mail processing protocols with popular list server software such as Majordomo [27] and Mailman [26].

9. ACKNOWLEDGEMENTS

We would like to thank Jim Basney and the anonymous reviewers for helpful comments and suggestions. This work was funded by the Office of Naval Research under contract numbers N00014-03-1-0765 and N00014-04-1-0562.

10. REFERENCES

[1] M. Abadi, N. Glew, B. Horne, B. Pinkas, " Certified Email with a Light On-line Trusted Third Party:Design and Implementation", in proceedings of the 11th International World Wide Web Conference, May 2002.

[2] M. Abadi, A. Birrell, M. Burrows, F. Dabek, and T. Wobber, "Bankable Postage for Network Services", in Proceedings of the 8th Asian Computing Science Conference, Mumbai, India, December 2003.

[3] J. Allen, *The CERT Guide to System and Network Security Practices*, Carnegie Mellon Software Engineering Institute, Addison Wesley, Indianapolis, 2001. ISBN 0-2-1-73723-X.

[4] I. Androutsopoulos *et al.*, "An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages", in Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Greece, July 2000.

[5] R. Anderson and R. Needham, "Robustness principles for public key protocols", in Advances in Cryptology (CRYPTO 95), 1995.

[6] M. Bellare, R. Canetti, and H. Krawczyk, "Message authentication using hash functions: The HMAC construction", RSA Laboratories' CryptoBytes, Vol. 2, No. 1, Spring 1996.

[7] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography", in Eurocrypt'98, LNCS 1403, Springer-Verlag, 1998.

[8] C. Blundo, S. Cimato, and R. D. Prisco, "Certified Email: Design and Implementation of a New Optimistic Protocol", in proceedings of the Eighth IEEE International Symposium on Computers and Communications, June 30 - July 03, Turkey, 2003.

[9] D. Boneh, X. Ding, G. Tsudik and B. Wong, "Fast Revocation of Security Capabilities", in Proceedings of the Usenix Security Symposium, August 2001.

[10] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing", *SIAM Journal of Computing*, Vol. 32, No. 3, pp. 586-615, 2003.

[11] Catalist, the official catalog of LISTSERV lists, *http://www.lsoft.com/catalist.html.*

[12] R. Clayton and M. Bond, "Experience Using a Low-Cost FPGA Design to Crack DES Keys", in Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2002.

[13] L. Cranor and B. LaMacchia, "Spam!", *Communications of the ACM* 41, 8 (August 1998), 74-83.

[14] X. Ding and G. Tsudik, "Simple Identity-Based Cryptography with Mediated RSA", in Proceedings of the RSA Conference, Cryptographer's Track, 2003.

[15] C. Dwork, M. Naor, "Pricing via Processing or Combatting Junk Mail", in Proceedings of CRYPTO'92, 1993, pp. 137–147.

[16] C. Dwork, A. Goldberg, and M. Naor, "On Memory-Bound Functions for Fighting Spam", in advances of Cryptology (CRYPTO 2003,) August 2003.

[17] E. Gabber *et al.*, "Curbing junk e-mail via secure classification", in Proceedings of Financial Cryptography, 1998.

[18] T. E. Gamal, "A Public Key Cryptosystem and a Signature Scheme Based on the Discrete Logarithm", *IEEE Transactions of Information Theory*, pages 31(4): 469-472, 1985.

[19] The GNU Privacy Guard, *http://gnupg.org.*

[20] J. Ioannidis, "Fighting spam by encapsulating policy

in email addresses", in Proceedings of the Symposium on Network and Distributed Systems Security, 2003.
[21] A. Ivan and Y. Dodis, "Proxy Cryptography Revisited", in Proceedings of the Network and Distributed System Security Symposium (NDSS), February 2003.
[22] Y. Kim, A. Perrig and G. Tsudik, "Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups", in Proceedings of 7th ACM Conference on Computer and Communication Security (CCS), 2000.
[23] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", IETF PEM WG RFC 21, 1993.
[24] LISTSERV, *http://www.lsoft.com.*
[25] T. Loder, M. V. Alstyne, and R. Wash, "An Economic Answer to Unsolicited Communication", in proceedings of the 5th ACM conference on Electronic Commerce, May 2004.
[26] Mailman, the GNU mailing list manager. *http://www.list.org.*
[27] Majordomo, *http://www.greatcircle.com/majordomo.*
[28] M. Mambo and E. Okamoto, "Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts", *IEICE Transactions on Fundamentals*, vol. E80-A, No. 1, 1997.
[29] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting", in proceedings of ACM SIGCOMM 1997.
[30] Ostermiller Java Utilities. *http://ostermiller.org/utils.*
[31] T. Tompkins and D. Handley, "Giving e-mail back to the users: Using digital signatures to solve the spam problem", First Monday, 8(9), September 2003.
[32] US Department of Energy Computer Incident Advisory, January 26 2000. *http://ciac.llnl.gov/ciac/bulletins/k-020.shtml.*
[33] C. K. Wong, M. G. Gouda, S. S. Lam, "Secure group communications using key graphs", *IEEE/ACM Transactions on Networking* 8(1): 16-30, 2000.
[34] P. Zimmerman, *The Official PGP User's Guide*, MIT Press, ISBN: 0-262-74017-6, May 1995.

APPENDIX

A. SELS ENCRYPTION SCHEME $\mathcal{E}$

Theorem 1 - Let $\mathcal{E} = (IGen, UGen, AEnc, ADec, \Gamma)$ be the SELS encryption scheme. $\mathcal{E}$ is CPA secure against the List Server and any Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$, if El Gamal is CPA secure against such adversaries.

Define,

$$Succ_{LS,\mathcal{E}} \stackrel{def}{=} \Pr\left[b = \hat{b} \;\middle|\; \begin{array}{l} (g,p,q,K_{LM},g^{K_{LM}},K_{LS},g^{K_{LS}}) \leftarrow IGen(1^k), \\ (K_u, K'_u) \leftarrow UserGen(1^k, K_{LS}, K_{LM}), \\ b \leftarrow \{0,1\}, (m_0,m_1) \leftarrow LS(g^{K_u}, K'_u), \\ \hat{b} \leftarrow LS(g^{K_u}, K'_u, Enc_{g^{K_u}}(m_b)) \end{array} \right]$$

Then $\mathcal{E}$ is CPA (Chosen Plaintext Attack) secure against LS if $|Succ_{LS,\mathcal{E}} - \frac{1}{2}|$ is negligible for LS. A similar formulation can be made for other adversaries with slight notational changes.

Proof: We have two types of adversaries to consider: the *List Server* (LS) and users outside the list. (List subscribers and the *List Moderator* (LM) receive emails from the sender, and, therefore, are not adversaries.) We first consider LS and assume that it can break the SELS encryption scheme $\mathcal{E}$. Then $|Succ_{LS,\mathcal{E}} - \frac{1}{2}|$ is non-negligible. Based on LS's algorithm to break $\mathcal{E}$, we create a probabilistic, polynomial time (PPT) algorithm $\mathcal{B}$ to mount a successful chosen plaintext attack against the standard El Gamal encryption scheme. However, our premise is that El Gamal is CPA secure. This fact will provide the contradiction to our assumption that LS can mount a successful CPA attack against $\mathcal{E}$.

We note that LS has knowledge of multiple corresponding private keys. Intuitively, we see that LS cannot decrypt any e-mails with these keys, and cannot gain any knowledge of the list key K_{LK} because its view of the corresponding private keys can be made consistent with any value of K_{LK}. However, we formally prove that LS cannot break $\mathcal{E}$ without breaking El Gamal. To prove this we use the idea that an adversary can simulate the role of LM and subscribers since all that LS receives from them are randomly chosen integers.

An oracle executes $IGen$ and $UGen$ with LS to generate private and corresponding private keys K_{LM}, K_{LS}, K_u, and K'_u. The oracle then gives $\mathcal{B}$ keys K_{LM} and g^{K_u} (where g^{K_u} is the El Gamal challenge public key). $\mathcal{B}$ then simulates the subscribing and unsubscribing of (polynomial many) users for LS by repeated execution of $UGen$ and sending of "unsubscribe" messages. LS now chooses two messages (m_0, m_1) to challenge the security of our encryption scheme $\mathcal{E}$. $\mathcal{B}$ considers these messages as the challenge to standard El Gamal and receives the challenge $Enc_{g^{K_u}}(m_b)$ from a left-right oracle. This $Enc_{g^{K_u}}(m_b) = AEnc_{g^{K_u}}(m_b)$ challenge is forwarded to LS who has a distinguisher able to determine b with probability greater than .5 by the assumption that $|Succ_{LS,\mathcal{E}} - \frac{1}{2}|$ is non-negligible. However, this is a contradiction to the assumption that El Gamal is CPA secure. Therefore, if El Gamal is CPA secure, our encryption system $\mathcal{E}$ is CPA secure against LS.

We now consider adversaries outside of the list. It is trivial to see that outsiders would have to break El Gamal to decrypt a message. This is because both messages from the sender to LS and messages from LS to the receivers are encrypted with valid El Gamal keys which are unknown in part or whole to any outsider. The formal proof is similar to the previous one. The main difference is that an outsider does not participate in the protocol at all. In fact, the algorithm $\mathcal{B}$ simply simulates an entire SELS system for the adversary giving him access to the communications and to all public-keys. At some point in time the adversary breaks $\mathcal{E}$, and we can show that $\mathcal{B}$ can break El Gamal for the same messages.