Journal of Network and Systems Management, Vol. 4, No. 3, 1996

Practical Protocols for Certified Electronic Mail

Robert H. Deng,[1] Li Gong,[2] Aurel A. Lazar,[3] and Weiguo Wang[1]

Electronic mail, or e-mail, has brought us a big step closer towards the vision of paperless offices. To advance even closer to this vision, however, it is essential that existing e-mail systems be enhanced with value-added services which are capable of replacing many of the human procedures established in pen and paper communications. One of the most important and desirable such services is certified e-mail delivery, in which the intended recipient will get the mail content if and only if the mail originator receives an irrefutable proof-of-delivery from the recipient. In this paper, we present the design of two third-party based certified mail protocols, termed CMP1 and CMP2. Both protocols are designed for integration into existing standard e-mail systems and both satisfy the requirements of *nonrepudiation of origin, nonrepudiation of delivery, and fairness*. The difference between CMP1 and CMP2 is that the former provides no mail content confidentiality protection while the latter provides such a protection. Moreover, security of the protocols are analyzed using a recently proposed accountability framework.

KEY WORDS: Digital signature; electronic mail; encryption; security.

1. INTRODUCTION

Electronic mail, or e-mail in short, is probably the most visible application of computer networking in use today. It is accessible to all users regardless of their level of technical expertise and it provides an automatic delivery service allowing users, separated by location and time, to exchange electronic messages. Modern e-mail systems handle not only text, but also electronic documents, voice, graphics, and financial transactions such as electronic data interchange.

[1] Institute of Systems Science, National University of Singapore, Singapore 119597. E-mail: [deng, wwang]@iss.nus.sg.
[2] SRI International, Computer Science Laboratory, Menlo Park, California 94025. E-mail: gong@csl.sri.com.
[3] Department of Electrical Engineering and Center for Telecommunications Research, Columbia University, New York, New York 10027-6699. E-mail: aurel@ctr.columbia.edu.

As such, e-mail systems are rapidly taking over as the medium of choice for the exchange of any kind of electronic information.

Indeed, e-mail has taken us a big step closer towards the vision of paperless offices. To advance even closer to this vision, however, it is essential that existing e-mail systems be enhanced with value-added services which are capable of replacing many of the human procedures established in pen and paper communications. One of the most important and desirable such services is certified e-mail delivery, in which the intended recipient will get the mail content if and only if the mail originator receives an irrefutable proof-of-delivery from the recipient which certifies the content of the mail [1]. This definition of certified mail is different from that used in the U.S. Postal Service, where certification of a letter is a marker that allows the letter to be traced.

The problem of certified mail delivery is in essence a problem of ''simultaneous'' exchange of a mail message and its proof-of-delivery between two parties that potentially do not trust each other: the mail originator would like to have a receipt from the intended mail recipient certifying reception of the mail content; at the same time, the recipient would like to make sure that his receipt is certifying the right mail content and that he will get the mail after issuing his receipt. Certified mail protocols based on simultaneous secret exchange (or gradual secret releasing) schemes have been studied for some time [see for examples, 1–8]. In simultaneous secret exchange schemes, it is assumed that two parties A and B each possess a secret a and b, respectively, where a and b are n bit strings. Further it is assumed that both secrets represent some value to the other party and that they are willing to trade the secrets with each other. A simultaneous secret exchange process is typically carried out as following. First, A and B exchange $f(a)$ and $g(b)$ for some predefined functions $f(\ )$ and $g(\ )$, with the property that A can not get b from $g(b)$ and B can not recover a from $f(a)$. Then, A and B release a and b bit-by-bit. For such a protocol to be useful, it must satisfy the following two requirements: *correctness*—the correctness of each bit given must be checked by each receiver to ensure that his/her secret has not being traded for garbage; and *fairness*—the computational effort required from the parties to obtain each other's remaining secret should be approximately equal at any stage during the execution of the protocol. Note that this fairness definition based on equal computational complexity makes sense only if the two parties have equal computing power, an often unrealistic and undesirable assumption [9]. Certified mail protocols based on simultaneous secret exchange are theoretically interesting since they enable two distrusted parties so simultaneously exchange a mail message and its receipt in the absence of a trusted third party (TTP). However, they have very limited practical value for reasons explained later.

Standard e-mail systems, such as Internet mail system [10] and the X.400 message handling system [11], have entered the mainstream of corporate life and many people today take use of these systems for granted to meet their daily communications needs. For certified mail to be useful and accepted by general users, it must be integrated into the existing standard e-mail systems. A common feature of these standard e-mail systems is that they use an underlying message transport architecture which is asynchronous and store-and-forward in nature. This implies that certified mail in such systems must also be asynchronous and store-and-forward in nature. Certified mail protocols based on simultaneous secret exchange require the existence of a real-time, interactive communication channel between the parties involved. Such a requirement obviously renders these protocols impractical to implement in standard e-mail systems.

In this paper, we are concerned with the design and analysis of practical certified mail protocols which can be integrated into existing standard e-mail systems. Physical certified mail delivery in postal systems is familiar with everyone, where the simultaneous mail and proof-of-delivery exchange between the message originator and recipient is mediated by a trusted postman. The postman is trusted by the originator to get the correct proof-of-delivery from the recipient and also trusted by the recipient to deliver the correct mail. Our protocols presented in this paper employ an electronic trusted postman (i.e., a TTP), and meet the requirements of *nonrepudiation of origin*, *nonrepudiation of delivery, and fairness* (to be defined more precisely later). In addition, fairness of these protocols depends only on the trustworthiness of the postman, and is independent of the computational powers of the involved parties. It should be noted that our protocols can be adapted for certified delivery of any type of information sent over asynchronous and store-and-forward networks. For example, sending fax over Internet is becoming popular and our protocols can be used to realize certified fax delivery with virtually no modifications.

The rest of the paper is organized as follows. In Section 2, we formulate a model for certified mail delivery and state precise requirements which should be met by our protocols. In Section 3, we give two certified mail protocols, called CMP1 and CMP2. Protocol CMP1 does not provide mail content confidentiality protection while CMP2 is an extension of CMP1 enhanced with mail content confidentiality protection capability. In Section 4, we analyze the two protocols using the accountability framework recently proposed by Kailar [12]. In Section 5, we conclude the paper with a summary of results and discussions for future work.

2. MODEL AND REQUIREMENTS

The network model adopted in this paper for certified mail delivery follows the generic messaging model as defined in the X.400 Message Handling System [11, 13] and the Internet mail system [10]. In this model, e-mail messages are transported through a message transfer system which is a store-and-forward network comprising one or more message transfer agents. At the edge of the model, a user agent acts on behalf of a user and interfaces to its local message transfer agent. To send an e-mail message from one user to another, the originating user employs a user agent to prepare the message and submit it to a local message transfer agent. To deliver the mail, a message transfer agent must decide if it can deliver the message directly to the recipient. If so, it will deliver the message to a user agent acting on behalf of the recipient user. If not, it contacts an adjacent message transfer agent, which is closer to the recipient, and negotiates transfer of the message. This store-and-forward process continues until the last message transfer agent in the sequence delivers the message to the destination user agent acting on behalf of the recipient user.

E-mail systems based on this model exhibit two important characteristics [10, 13]: (1) mail transfer is store-and-forward in nature; (2) the user agents for the originator and recipient need not be ''on-line'' simultaneously for message to be submitted, transported, and delivered. In fact, only the node currently responsible for the electronic mail message, and the ''next hop'' taking responsibility for the message, need be connected in order for the message to be transferred. This asynchronous nature of e-mail system is extremely useful in business environments, because it allows information to be exchanged without requiring a simultaneous session between two parties [13].

Entities involved in our certified mail model and the assumptions made on each of them are the following:

- *The Network.* The network is responsible for mail transfer among users and the trusted postman. It is neither secure nor reliable (e.g., Internet). Mail sent over the network can be read and intercepted by anyone with the right equipment, and mail may also be lost or mis-routed due to network fault or system crash. We do assume however, that fault/crash recovery period is shorter than the maximum mail delivery interval, which can be up to fifteen days in Internet.
- *Public Key Certificate Authority (CA).* All certified mail protocols presented in this paper use a public key cryptosystem [14]. As a result, every active party in the model is associated with a pair of public/private keys. The CA certifies a public key of party X by digitally signing a data structure consisting of the identity of X and X's public key.
- *User.* A user is equipped with cryptographic processing capability to encrypt/decrypt messages, and to sign and verify digital signatures. Since a user agents acts on behalf of a user, in the rest of the paper we will use the terms user and user agent synonymously.
- *Postman.* The postman is an automatic process which plays more or less the same role as a real postman in a postal system. It is trusted to faith-

fully deliver a certified mail to the correct recipient in exchange for a proof-of-delivery from the recipient. The postman can be implemented on a secure fault tolerant architecture such as Rampart [15].

Based on this model and assumptions, our certified mail protocols should meet the following requirements.

- *Nonrepudiation of origin.* Providing the recipient of an e-mail with irrefutable proof that the mail content received was the same as the one sent by the originator. This proof-of-origin can protect against any attempt by the originator to falsely deny sending that message.
- *Nonrepudiation of delivery.* Providing the mail originator with irrevocable proof that the mail content received by the recipient was the same as the one sent by the originator. This proof-of-delivery can protect against any attempt by the recipient to falsely deny receiving the message.
- *Fairness.* Proper execution of the protocol ensures that the proof-of-delivery from the mail recipient and the proof-of-origin from the mail originator are available to the mail originator and recipient, respectively. Moreover, the protocol must be fail-safe. That is, incomplete execution of the protocol will not result in a situation where the proof-of-delivery is available to the originator but the proof-of-origin is not available to the recipient, or vice versa.

3. THE PROTOCOLS

There are three active parties involved in the certified mail delivery system: the mail originator *A* (Alice), the mail recipient *B* (Bob), and the postman *PM*. The protocols do not require that a simultaneous session be established between users. Protocol messages are sent simply as ordinary e-mail messages. A guiding principle in our protocol design is to minimize the number of message exchanges among users and the postman. Therefore, in this section we first examine various protocol message flow scenarios among the three parties and then identify the optimal one in the sense that it requires the minimum number of protocol message exchanges to accomplish the job. Based on this optimal scenario, we then present two certified mail protocols which satisfy the three requirements listed in the last section.

3.1. An Optimal Scenario

There are various ways to arrange message exchanges among *A*, *B*, and *PM*. One possible approach is illustrated in Fig. 1a. In this scenario, a dispatch

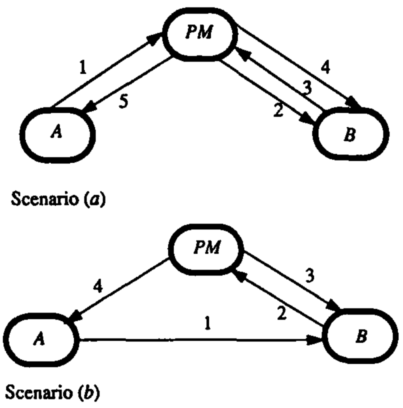

Fig. 1. Two scenarios of protocol message exchanges in certified mail delivery.

a mail content *m* in a certified mail from *A* to *B*, *A* sends the certified mail which contains *m* and *A*'s proof-of-origin to *PM* in protocol message 1. Upon receiving message 1, *PM* sends protocol message 2 to *B* telling him that "You have a certified mail from *A*. Please send in your proof-of-delivery if you want to receive it." *B* responds to *PM* by sending his proof-of-delivery in protocol message 3. *PM* then transfers *m* and *A*'s proof-of-origin to *B* in protocol message 4 and delivers *B*'s proof-of-delivery to *A* in protocol message 5.

Scenario in Fig. 1a requires 5 protocol messages. Before studying ways of improving this approach, let us first consider the minimum number of protocol messages needed in a certified mail protocol. The mail originator *A* needs to send at least one protocol message (the certified mail); the mail recipient *B* needs to send at least one protocol message containing his proof-of-delivery; and since the postman *PM* functions as a mediator in the exchange of *A*'s mail content *m* and *B*'s proof-of-delivery, *PM* needs to send at least two protocol messages, one to *A* and one *B*. Thus, a scenario is optimal if it requires only 4 protocol messages. Scenario in Fig. 1a is obviously suboptimal.

An improved version is shown in Fig. 1b. Here *A* sends the certified mail directly to *B* in protocol message 1. In order to prevent *B* from reading the mail content *m* without handling in his proof-of-delivery, the mail content is encrypted under a session key known to *PM* but not known to *B*. Upon reception of protocol message 1, *B* sends protocol message 2 to *PM*, which consists of the encrypted mail content and *B*'s proof-of-delivery. *PM* then decrypts the encrypted mail content, sends the mail content *m* to *B* in protocol message 3 and sends *B*'s proof-of-delivery to *A* in protocol message 4. This scenario is optimal and serves as a starting point in our design of certified mail protocols.

The certified mail protocols described in this paper all use public key cryptosystems, where each party has a public/private key pair. It is assumed that a party's public key is made available to other parties in an authenticated manner and is outside the scope of the present paper.

This description is only a high level overview of operations of certified mail protocols and we have purposely left some important design issues unaddressed. For example, it ignores the problem of unreliable message delivery. We will tackle these in Sections 3.3 and 3.4.

3.2. Notations

In all protocol descriptions, we will make use of the following notations:

SK_P:	private key of party P, used for signing digital signatures
PK_P:	public key of party P, used for encryption and for verifying signatures signed under SK_P
$h(x)$:	output of an one-way hash function $h(\ )$ with message x as its input
$\{x\}PK_P$:	encryption of message x under P's public key PK_P
$\{x\}SK_P$:	message x signed with P's private key SK_P
$[x]k$:	symmetric-key encryption of message x under a session key k

3.3. CMP1: A Protocol without Message Content Confidentiality Protection

In both CMP1 and CMP2 we assume that each party can obtain and verify the validity of the public keys of other parties, either from a public key certificate directory or by local cashing public key certificates. We further assume that only approximate clock synchronization is possible between communicating parties. This is a reasonable assumption since it is unrealistic to assume that synchronous clocks be maintained in a large distributed e-mail system and since the asynchronous nature of e-mail service may need to tolerate clock drift up to minutes or even hours.

A concise representation of protocol message flows in CMP1 is provided in Fig. 2, where it is assumed that appropriate timestamps (not shown in the figure) are included in all signed messages.

To send a mail message containing m to B using certified mail, A first digitally signs $\{A, B, PM, m\}$ with her private key to produce $\{A, B, PM, m\}SK_A$. A then generates a session key k and encrypts the signed data under k using a symmetric key cryptosystem. Finally, A computes $h(m)$ and sends protocol message 1 (i.e., the certified mail) to B. The clear text part (i.e., A, B, PM, $h(m)$) in this message serves as the mail identifier. It informs B that "You

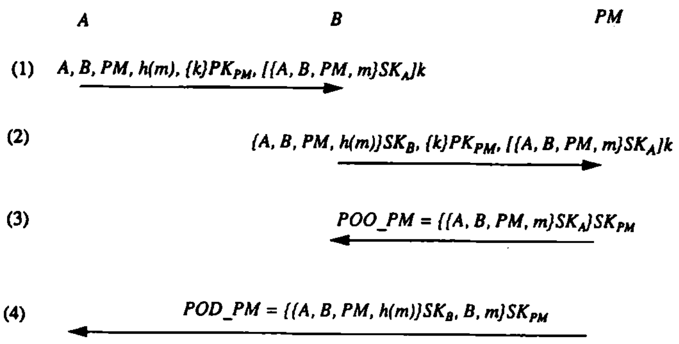
A
B
PM
(1)
A, B, PM, h(m), {k}PKPM, [{A, B, PM, m}SKA]k
(2)
{A, B, PM, h(m)}SKB, {k}PKPM, [{A, B, PM, m}SKA]k
(3)
POO_PM = {{A, B, PM, m}SKA}SKPM
(4)
POD_PM = {{A, B, PM, h(m)}SKB, B, m}SKPM

Fig. 2. Protocol CMP1.

have a certified mail from A waiting for you. Please contact PM if you wish to receive it''.

After receiving protocol message 1, B is faced with two choices. He may just ignore the message if he is not interested in receiving the message from A. In this case, the protocol is aborted. If B chooses to receive the message, he signs $\{A, B, PM, h(m)\}$ using his private key and sends protocol message 2 to PM. Note that this message contains $\{A, B, PM, h(m)\}SK_B$ as well as the encrypted parts, $\{k\}PK_{PM}$ and $[\{A, B, PM, m\}SK_A]k$, of protocol message 1.

Upon receiving message 2, PM first checks the validity of $\{A, B, PM, h(m)\}SK_B$ using the public key of B. It then decrypts $\{k\}PK_{PM}$ using its private key, and decrypts $[\{A, B, PM, m\}SK_A]k$ using k. Next, PM checks the validity of $\{A, B, PM, m\}SK_A$ using A's public key, computes $h(m)$, and compares this $h(m)$ with the one received in $\{A, B, PM, h(m)\}SK_B$. If the two values match, PM accepts that m is the mail content A wanted to send to B and that B is willing to receive m. In this case, PM computes protocol message 3

$$POO\_PM = \{\{A, B, PM, m\}SK_A\}SK_{PM}$$

and protocol message 4

$$POD\_PM = \{\{A, B, PM, h(m)\}SK_B, B, m\}SK_{PM}$$

POO_PM is A's proof-of-origin and POD_PM is B's proof-of-delivery, both being notarized by PM.

After generating protocol messages 3 and 4, PM must make sure that message 3 (i.e., POO_PM) be received by B and message 4 (i.e., POD_PM) be received by A. One approach to accomplish this task is to have PM send message 3 (4) to B (A), starts up a timer, and waits for an ACK (acknowledge) from B (A). If the ACK is not received when the timer expires, PM retransmits message 3 (4). This process continues until the message is acknowledged by B

(*A*). This approach ensures reliable delivery of messages 3 and 4 to *B* and *A*, respectively, over unreliable networks. However, it suffers from the so called *reluctant receiver* problem. After *B* getting message 3 and therefore read the mail content *m*, he may purposely refuses to hand in his *ACK* to *PM*. Without getting the *ACK*, *PM* is not sure whether *B* has received message 3 and this may result in a situation unfair to *A*.

A better approach is to have *A* and *B* poll *PM* for messages 4 and 3, respectively. *A* and *B* must continue to poll *PM* if the respective messages are not received. Note that it is *A*'s (*B*'s) responsibility to get message 4 (3). Therefore, this method does not suffer from the reluctant receiver problem.

Yet another approach is to have *PM* maintain a public bulletin board. Messages published in the bulletin board are public information and can be read by everyone. However, only *PM* has the privilege of writing messages to and erasing messages from the bulletin board. Our protocol may take advantage of this public bulletin board and works as follows. After generating protocol messages 3 and 4, *PM* sends message 3 to *B* and sends message 4 to *A*. The messages are sent only once, there are no retransmissions. *PM* then publishes messages 3 and 4 in the bulletin board. Under normal network operations, *B* will receive message 3 and *A* will receive message 4. In this case, the bulletin board is not accessed by *A* and *B*. Only when *B* (*A*) does not receive message 3 (message 4) within a pre-defined time interval will *B* (*A*) fetch message 3 (message 4) from the bulletin board. This can be done, for examples, through *Mosaic*, *Netscape*, or *ftp* in the case users are connected over Internet. We remark here that messages should be kept in the bulletin board for a sufficient long period of time (say one or two weeks) so that users are able to read these messages even the network fails temporarily.

From the protocol description it is clear that, upon successful completion of the protocol, *A* is in possession of *POD_PM* and *B* is in possession of *POO_PM*. In case of dispute, *A* may present *POD_PM* to a judge to prove that *B* has received the message content *m* and *B* may present *POO_PM* to the judge to show that *A* is the person who has sent the message *m*. A detailed analysis on the accountability of *POO_PM* and *POD_PM* is given in Section 4.

3.4. CPM2: A Protocol with Message Content Confidentiality Protection

In protocol CMP1, the mail content *m* is embedded in *POD_PM* (message 4) and *POO_PM* (message 3) and the clear texts of these two messages are sent over the network and/or published in the bulletin board. Therefore, protocol CMP1 does not provide any confidentiality protection on *m*. Certified mail is quite often used to deliver sensitive or classified information, such as income tax assessments, credit card numbers and corporate proprietary information.

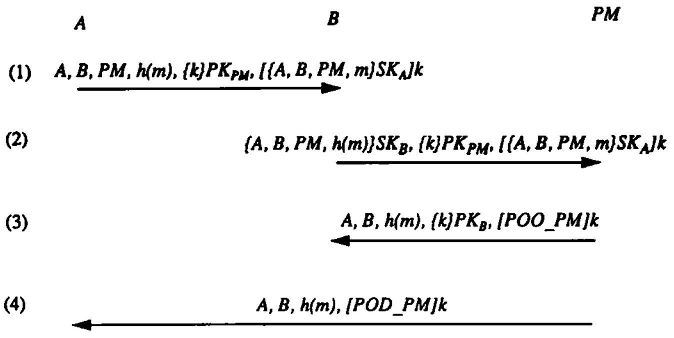
A
B
PM
(1) A, B, PM, h(m), {k}PK_PM, [{A, B, PM, m}SK_A]k
(2) {A, B, PM, h(m)}SK_B, {k}PK_PM, [{A, B, PM, m}SK_A]k
(3) A, B, h(m), {k}PK_B, [POO_PM]k
(4) A, B, h(m), [POD_PM]k

Fig. 3. Protocol CMP2.

The contents of this information are lucrative targets for eavesdropping, and therefore must be protected against disclosure to outsiders.

CMP2, as shown in Fig. 3, is a simple modification to CMP1. In Fig. 3, protocol messages 1 and 2 are identical to protocol messages 1 and 2 in CMP1, respectively; while protocol messages 3 and 4 are encrypted versions of messages 3 and 4 in CMP1, respectively. The encryptions prevent the mail content m from being disclosed to outsiders. Specifically, in protocol message 3, $POO\_PM = \{\{A, B, PM, m\}SK_A\}SK_{PM}$ is encrypted with the session key k (which is the same session key used in protocol message 1) and k is encrypted with the public key of B. In protocol message 4, $POD\_PM = \{\{A, B, PM, h(m)\}SK_B, B, m\}SK_{PM}$ is encrypted with the same session key k. Note that, since k has been used by A to encrypt protocol message 1, A has knowledge of k. Hence, an encrypted version of k (i.e., $\{k\}PK_A$) need not be included in protocol message 4. In both protocol messages 3 and 4, the message headers A, B, $h(m)$ are kept in clear. They serve as an index to the certified mail and can be used by A and B in retrieving messages 4 and 3, respectively, from PM. Operation of CMP2 is identical to that of CMP1, except the added symmetric key encryption and decryption, and therefore, will not be described here.

4. SECURITY ANALYSIS

Security analysis of CMP1 and CMP2 follow almost the same procedures. Therefore, in this section we will only show the security analysis of CMP1. We first present an analysis of CMP1 using the accountability framework of [12]. In the course of this analysis, we formalize various assumsptions underlying CMP1 and prove that the protocol satisfies the *nonrepudiation of origin* and *nonrepudiation of delivery* requirements. We then argue in an informal manner that the protocol also satisfies the *fairness* requirement.

4.1. Accountability Analysis

In this subsection we analyze CMP1 using the accountability framework proposed by Kailar [12]. For a brief overview of the framework and the meaning of constructs and postulates used in our analysis, the reader is directed to the Appendix provided at the end of the paper. The main objective of this analysis is to prove that CMP1 satisfies the *nonrepudiation of origin* and *nonrepudiation of delivery* requirements. Using the constructs of the accountability framework, this is equivalent to show that the protocol meets the following goals:

G1: A *CanProve* (B *received* m)
G2: B *CanProve* (A *sent* m)

First, we need to interpret the protocol messages (see Fig. 2) using the constructs of the accountability framework. In the analysis, only those messages which are signed and have plain-text contents have accountability, and need be interpreted. Therefore, protocol message 1 will not be interpreted. Protocol messages 2, 3, and 4 are interpreted as

2. PM *Receives* h(m) *SignedWith* SK_B; PM *Receives* m *SignedWith* SK_A
3. B *Receives* (m *SignedWith* SK_A) *SignedWith* SK_{PM}
4. A *Receives* (h(m) *SignedWith* SK_B, B, m) *SignedWith* SK_{PM}

respectively. The initial assumptions that are required in the analysis are listed next:

A1. A, B *CanProve* (PK_{PM} *Authenticates* PM)
A2. A, PM *CanProve* (PK_B *Authenticates* B)
A3. B, PM *CanProve* (PK_A *Authenticates* A)
A4. A, B *CanProve* (PM *IsTrustedOn* (PM Says))
A5. (A *Says* m) => (A *sent* m)
A6. (B *Says* h(m)) => (B *received* h(m))
A7. (PM *Says* (B, m)) => (PM *Says* m *has been successfully sent to* B)
A8. (B *Received* h(m)) ∧ (m *has been successfully sent to* B) => B *Received* m

Assumption A1 states that *A* and *B* can prove that *PM* can be authenticated by the key PK_{PM}. That is, *A* and *B* can prove that *PM* is accountable for any message signed with SK_{PM}. This assumption is justified if either PK_{PM} is known to authenticate *PM* globally, or if *A* and *B* can acquire and present a public key certificate for PK_{PM} issued by the CA (see Section 2). Assumptions A2 and A3 are similar to A1. They state, in effect, that the association of parties to statements can be proved using public key certificates. Assumption A4 is justified since *PM* is a TTP which is globally trusted. Assumption A5 is about implication of what *A* states in message 3, and the last three assumptions are about implications of what *B* and *PM* state in message 4. The analysis of the protocol is carried out as following on a message by message basis.

Message 2: This message is equivalent to

2.1 PM *Receives* h(m) *SignedWith* SK_B;
2.2 PM *Receives* m *SignedWith* SK_A

When *PM* receives message 2.1, using the assumption that *PM* can prove the association of the signature on message 2.1 with *B* (i.e., assumption A2), and applying the **Signature** postulate,

PM *CanProve* (B *Says* h(m))

This statement can be refined using assumption A6 and the **Inference** postulate, as

PM *CanProve* (B *Received* h(m))

When *PM* receives message 2.2, using the assumption that *PM* can prove the association of the signature on message 2.2 with *A* (i.e., assumption A3), and applying the **Signature** postulate,

PM *CanProve* (A *Says* m)

This statement can be further refined using assumption A5 and the **Inference** postulate as

PM *CanProve* (A *sent* m)

At this point, *PM* computes *h*(*m*) with *m* as the input, and compares this value with the hash function output received in message 2.1. If the two match, *PM* is confident that *A* has sent the message *m*, and *B* has committed himself in receiving the message *m*.

Message 3: When *B* receives message 3, using the assumption that *B* can prove the association of *PM* with *PM*'s signature (i.e., assumption A1), and applying the **Signature** postulate,

B *CanProve* (PM *Says* (m *SignedWith* SK_A)

Using assumption A4 about the trust on *PM*, and applying the **Trust** postulate on the result derived above, we have

B *CanProve* (m *SignedWith* SK_A)

Using the assumption that *B* can prove that PK_A authenticates *A* (i.e., assumption A3), and applying the **Signature** postulate again, we have

B *CanProve* (A *Says* m)

This statement can be refined using assumption A5 and the **Inference** postulate, as

B *CanProve* (A *sent* m) [G1]

Hence, by presenting message 3 (i.e., *POO_PM*) to a judge, *B* can prove that *A* sent him the message *m*. This completes the proof that CMP1 meets the *nonrepudiation of origin* requirement.

Message 4: Message 4 is equivalent to

4.1 A *Receives* (h(m) *SignedWith* SK_B) *SignedWith* SK_{PM}
4.2 A *Receives* (B, m) *SignedWith* SK_{PM}

When *A* receives message 4.1, using assumptions A1, A4, and A6, and applying the **Signature, Trust,** and **Inference** postulates, by following a similar procedure as in the analysis of message 3, we have

A *CanProve* (B *Received* h(m))

When *A* receives message 4.2, using the assumption that *A* can prove the association of *PM* with *PM*'s signature (i.e., assumption A1), and applying the **Signature** postulate,

A *CanProve* (PM *Says* (B, m))

Using assumption A7 and the **Inference** postulate, is can be refined as

A *CanProve* (PM *Says* m *has been successfully sent to* B)

Using the assumption about the trust on *PM* (i.e., assumption A4), and applying the **Trust** postulate on this, we have

A *CanProve* (m *has been successfully sent to* B)

Using the two results, A *CanProve* (B *Received* h(m)) and A *CanProve* (m *has been successfully sent to* B), obtained earlier, and applying the **Conjunction** postulate, we get

A *CanProve* ((B *Received* h(m)) $\wedge$ (m *has been successfully sent to* B))

This result can be refined, using assumption A8 and applying the **Inference** postulate, as

A *CanProve* (B *Received* m) [G2]

That is, by presenting message 4 (i.e., *POD_PM*) to a judge, *A* can prove that *B* received message *m*. This completes the proof that CMP1 satisfies the *nonrepudiation of delivery* requirement.

4.2. Fairness Analysis

By our fairness definition given in Section 2, CMP1 meets the fairness requirement if successful completion of the protocol ensures that the mail originator A gets the proof-of-delivery *POD_PM* and that the mail recipient B gets the proof-of-origin *POO_PM*, and if incomplete execution of the protocol will not result in a situation which is unfair to either A or B.

We now claim that the fairness of CMP1 is guaranteed by the correct functioning of the postman *PM*. In sending protocol message 1 to B, A trusts *PM* to response to her with B's proof-of-delivery. On the other hand, in sending protocol message 2 to *PM*, B trusts *PM* to check the validity of A's encrypted message in protocol message 1 and to response to him with A's proof-of-origin. Specifically, After receiving protocol message 2, *PM* decrypts $[\{A, B, PM, m\}SK_A]k$, verifies the validity of $\{A, B, PM, m\}SK_A$ and $\{A, B, PM, h(m)\}SK_B$ using the public keys of A and B, respectively. *PM* then computes $h(m)$ with m as the hash function input, and compares this value with the received one. If A's signature on $\{A, B, PM, m\}SK_A$ or B's signature on $\{A, B, PM, h(m)\}SK_B$ can not be verified, or the newly computed $h(m)$ does not math the received one, *PM* terminates the protocol. As a result, if A cheats in protocol message 1, she will not get B's proof-of-delivery, and if B cheats in protocol message 2, he will not get A's proof-of-origin.

Note that use of message 3 $\{\{A, B, PM, m\}SK_A\}SK_{PM}$ instead of $\{A, B, PM, m\}SK_A$ as proof-of-delivery gives B explicit assurance that the m in message 3 is the one he has committed himself to receive in message 2. Also, note that use of message 4 $\{\{A, B, PM, h(m)\}SK_B, B, m\}SK_{PM}$ instead of simply $\{A, B, PM, h(m)\}SK_B$ protects the interest of A. The binding among m, B's identity, and B's signature in $\{\{A, B, PM, h(m)\}SK_B, B, m\}SK_{PM}$ is notarized by *PM*, and this binding gives A explicit assurance that B's signature is an non-refutable proof of B having received m.

Obviously, if, for whatever reasons, the protocol stops at any point between protocol messages 1 and 2, A will not have B's proof-of-delivery and B will not have A's proof-of-origin. An unfair situation to A results only if protocol message 3 reaches B but protocol message 4 fails to reach A, and an unfair situation to B results only if protocol message 4 reaches A but protocol message 3 fails to reach B. Therefore, it is important to have protocol messages 3 and 4 reach B and A, respectively and ''simultaneously''. Failure in sending these two messages may be due to fault in the mail transferring network, crash of A, B, or *PM*. To recover from system crashes, it is essential that each system write its messages into a stable memory, so that when the system comes back it can pick up from where it left. Since in our protocols A and B poll *PM* until they receive messages 4 and 3 respectively, we assume that the messages are maintained by *PM* for a relatively long period of time (longer than system fault/crash recovery time) so that A and B can get around of the fault/crash period to poll messages from PM.

5. SUMMARY AND FUTURE RESEARCH

We have presented the design and the analysis of two protocols, CMP1 and CMP2, for certified mail delivery. Both protocols are optimal in the number of protocol message flows and both meet the requirements of *nonrepudiation of origin*, *nonrepudiation of delivery, and fairness* as defined in Section 2. The only difference between CMP1 and CMP2 is that the former provides no mail content confidentiality protection while the latter provides such protection at the cost of performing some additional encryption and decryption operations. Previous work on certified mail protocols based on simultaneous secret exchange schemes require the existence of a realtime, interactive communication channel between the mail message originator and recipient. In contrast, protocols proposed in this paper are trusted third party based, and are designed for practical implementation in asynchronous and store-and-forward networks, such as Internet.

In a large distributed e-mail system, postman or TTP services for certified mail delivery may be provided by a wide spectrum of organizations. As the number of TTPs increases and as the system spans organizational boundaries, it may become difficult for users to assess the appropriate level of trust to place in the TTPs. When trust or confidence is lacking in the real world, one relies on endorsements, licensing, insurance, and surety bonds to compensate. By incorporating such assurance into a distributed system, users are better able to evaluate the risks incurred when using a particular TTP [16].

Protocols studied in the present paper all assume the existence of a single TTP which is trusted and accepted by both the mail originator and recipient. Therefore, the security and the competence of the single TTP is central to the security and the correct functioning of these centralized protocols. A compromised TTP may be biased towards either the mail originator or the mail recipient, and a faulty TTP may misbehave in an unpreditable manner or may render the system to a complete halt. A robust system should avoid as much as possible a single focal point in the system and instead distribute its trusts to multiple points. Another problem with centralized certified mail protocols is that, in certain situations, it may only be possible for a mail originator and a recipient to agree on a common set of TTPs, but impossible for them to settle down on any single TTP. Therefore, a very interesting and practical problem for future work is to design distributed certified mail protocols in which multiple TTPs share the responsibility of providing certified mail service and a minority of corrupted TTPs can not compromise the service through malicious behavior and collusion. Previous work on a distributed authentication protocol [17] and a distributed secure auction protocol [18] may be adapted for distributing security functionality to multiple TTPs.

APPENDIX A

In this appendix, we overview various constructs used in the accountability framework and list those postulates which are used in our accountability analysis in Section 4.1. For a complete description of the framework the reader is refereed to the original paper [12].

A.1. Definition of Accountability

Accountability is the property whereby the association of a unique originator with an object (e.g., electronic message) or action (e.g., electronic transaction) can be proved to a third party (i.e., a party who is different from the originator and the prover).

A.2. Constructs

''A *CanProve* x''—Principal A can prove statement x if, for any principal B, A can execute a sequence of operations such that after the sequence of operations, it has proved x to B without revealing any secret y ($x \neq y$) to B.

''PK *Authenticates* A''—This construct is used to denote the fact that key PK can be used to authenticate the signature of principal A, or to associate A unambiguously with any statement digitally signed with SK. It is used here to interpret public key certificates issued by certifying authorities.

''x *in* m''—x is an interpretation of a field, or an interpretation of a combination of fields in message m. This interpretation is protocol specific, and is expected to be defined by the protocol designers explicitly.

''A *Says* x''—Principal A is accountable for making the statement x, and anything that x implies. It is used for interpreting provability results of the form A *CanProve* (B *Says* x). Note that a principal A cannot prove that B Says x if x is a signed message field that is encrypted using a shared key. That is, if x has been encrypted for confidentiality using a key shared between A and B, then A cannot prove that B *Says* x. This follows from the definition of the *CanProve* construct, which says that in order to prove accountability, the prover should not have to reveal secrets (which are different from x) to the audience.

"A *Receives* m *SignedWith* SK"—Principal *A* receives message *m* which is signed with *SK*.

"A *IsTrustedOn* x"—Principal *A* is trusted on statement *x*. In particular, *A* has the authority to endorse statement *x*, and is liable for making statement *x*.

A.3. Postulates

Conjunction

$$\frac{\text{A } \textit{CanProve} \text{ x; A } \textit{CanProve} \text{ y}}{\text{A } \textit{CanProve} \text{ (x} \wedge \text{y)}}$$

That is, if *A* can prove that *x* holds and *A* can prove that *y* holds, then *A* can prove that $x \wedge y$ holds.

Inference

$$\frac{\text{A } \textit{CanProve} \text{ x; x} \Rightarrow \text{y}}{\text{A } \textit{CanProve} \text{ y}}$$

That is, if *A* can prove that *x* holds, and if *x* implies *y*, then it follows that *y* holds. Here the statement $x => y$ (*x* implies *y*) is used to articulate the interpretations of signed messages. Such interpretations are assumed to be defined explicitly by protocol designers, and hence, are assumed to be evident to all principals involved.

Signature

$$\frac{\begin{array}{c}\text{A } \textit{Receives} \text{ (m } \textit{SignedWith} \text{ SK); x } \textit{in} \text{ m;}\\ \text{A } \textit{CanProve} \text{ (PK } \textit{Authenticates} \text{ B)}\end{array}}{\text{A } \textit{CanProve} \text{ (B } \textit{Says} \text{ x)}}$$

That is, if *A* receives a message *m* which is signed with key *SK*, the message *m* contains statement *x*, and if *A* can prove that the key *PK* authenticates principal *B* or that *PK* authenticated *B* at the time the message was signed, then *A* can prove that *B* indeed says *x*.

Trust

$$\frac{\begin{array}{c}\text{A } \textit{CanProve} \text{ (B } \textit{Says} \text{ x);}\\ \text{A } \textit{CanProve} \text{ (B } \textit{IsTrustedOn} \text{ x)}\end{array}}{\text{A } \textit{CanProve} \text{ x}}$$

That is, if *A* can prove that *B*, who is an authority on *x*, says *x*, then *A* can prove that *x* holds.

REFERENCES

1. S. Even, O. Goldreich, and A. Lempel, A randomized protocol for signing contracts, *Communications of the ACM*, Vol. 28, pp. 637-647, June 1985.
2. E. F. Brickell, D. Chaum, I. Damgard, and J. Van de Graaf, Gradual and veriable release of a secret, *Advances in Cryptology*—CRYPTO'87, pp. 156-166, 1987.
3. M. Blum, How do exchange (secret) keys, Proceedings of STOC'83, pp. 440-447, 1983.
4. R. Cleve, Controlled gradual disclosure schemes for random bits and their applications, *Advances in Cryptology—CRYPTO'89*, pp. 573-588, 1989.
5. I. Damgard, Practical and provably secure exchange of digital signatures, *Advances in Cryptology—EUROCRYPT'93*, pp. 200-217, 1993.
6. M. Luby, S. Micali, and C. Rackoff, How to simultaneously exchange secret bit by flipping a symmetrically-biased coin, Proceedings of FOCS'83, pp. 23-30, 1983.
7. T. Okamoto and K. Ohta, How to simultaneously exchange secrets by general assumptions, Proceedings of 2nd ACM Conference on Computer and Communications Security, pp. 184-192, Fairfax, Virginia, November 1994.
8. A. Yao, How to generate and exchange secrets, Proceedings of FOCS'86, pp. 162-167, 1986.
9. M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest, A fair protocol for signing contracts, *IEEE Transactions on Information Theory*, Vol. 36, pp. 40-46, January 1990.
10. M. T. Rose, *The Internet Message: Closing the Book with Electronic Mail*, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1993.
11. CCITT, Message Handling Systems, X.400 Series Recommendations, 1988.
12. R. Kailar, Reasoning about accountability in protocols for electronic commerce, Proceedings of 1995 IEEE Symposium on Security and Privacy, pp. 236-250, Oakland, California, May 1995.
13. S. Radicati, *Electronic Mail: An Introduction to the X.400 Message Handling Standards*, McGraw-Hill, Inc., New York, 1992.
14. W. Diffie and M. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, Vol. 22, pp. 644-654, 1976.
15. M. Reiter, A secure group membership protocol, Proceedings of the Symposium on Research in Security and Privacy, pp. 176-189, Oakland, California, May 1994.
16. C. Lai, G. Medvinsky, and B. C. Neuman, Endorsements, licensing, and insurance for distributed system services, Proceedings of 2nd ACM Conference on Computer and Communications Security, pp. 170-175, Fairfax, Virginia, November 1994.
17. L. Gong, Increasing availability and security of an authentication service, *IEEE J. Selected Areas Communications*, Vol. 11, pp. 657-662, June 1993.
18. M. K. Franklin and M. K. Reiter, The design and implementation of a secure auction service, Proceedings of 1995 IEEE Symposium on Security and Privacy, pp. 2-14, Oakland, California, May 1995.

Robert H. Deng received his B.E. degree in electrical engineering from Changsha Institute of Technology, Changsha, China in 1981 and the M.S. and Ph.D degrees in electrical engineering from Illinois Institute of Technology, Chicago in 1983 and 1986, respectively. He is now a Research Staff Member at the Institute of Systems Science, National University of Singapore. He has more than 60 publications in the areas of digital communications, error control coding, network interconnection, network management and network security. He currently leads several projects in cryptographic techniques and information security.

Li Gong is a Computer Scientist at SRI International, where his current research interest is in distributed systems and communication networks, particularly in issues of fault tolerance and security. He received the B.S. degree (with honors) and the M.S. degree from Tsinghua University, Beijing, in 1985 and 1987 respectively, and the Ph.D. degree from the University of Cambridge (Jesus College), England, in 1990, all in computer science. He served as Program Co-Chair of the Third ACM Conference on Computer and Communications Security (1996) and Program Chair of the 7th and 8th IEEE Computer Security Foundations Workshops (1994 and 1995). He is also on the editorial board of the Journal of Computer Security. He received the IEEE Communications Society Leonard A. Abraham Prize Paper Award in 1994 and the IEEE Symposium on Security and Privacy Outstanding Paper Award in 1989.

Aurel A. Lazar is a Professor of Electrical Engineering at Columbia University since 1988. His research interests span both theoretical as well as experimental studies of telecommunication networks and intelligent systems. The theoretical research he conducted during the 80's pertains to the modeling, analysis and control of telecommunication networks and intelligent systems. He formulated optimal flow and admission control problems and, by building upon the theory of point processes, derived control laws for Markovian queueing network models in a game theoretic setting. He was the chief architect of two experimental networks, generically called MAGNET. This work lead to the concept of Asynchronous Time Sharing in realtime scheduling. Professor Lazar is the Director of the Telecommunication Networks Laboratory of the Center for Telecommunications Research, area editor for Network Management and Control for the IEEE Transactions on Communications, member of Verlag monograph series on Telecommunication Networks and Computer Systems.

Weiguo Wang received his B.E. degree in computer science and technology from the University, of Science and Technology of China, Hefei, China in 1983 and his M.A. and Ph.D degrees in Computer Science from Boston University, Boston, Massachusetts, in 1985 and 1990, respectively. He has been with the Institute of Systems Science, National University of Singapore since November 1990. His current research interest is multimedia networking architecture, network service creation and management, quality of service guarantees and secure and trusted network services.