Design and Implementation of an Inline Certified E-mail Service

Stelvio Cimato[1], Clemente Galdi[2], Raffaella Giordano[3], Barbara Masucci[2], and Gildo Tomasco

[1] Dipartimento di Tecnologie dell'Informazione, Universitá di Milano, Via Bramante 65, 26013 Crema (CR), Italy
cimato@dti.unimi.it

[2] Dipartimento di Informatica ed Applicazioni, Universitá di Salerno, Via S. Allende, 84081 Baronissi (SA), Italy
clegal, masucci@dia.unisa.it

[3] Italsime s.p.a. Via Cinthia 25, Parco S. Paolo, 80126, Napoli (NA), Italy
giordano.r@italsime.it

Abstract. Nowadays, e-mail has become one of the most widely used communication medium. Because of its characteristics of inexpensivity and rapidity in the delivery of messages, e-mail is increasingly used in place of ordinary mail. However, the e-mail service exposes users to several risks related to the lack of security during the message exchange. Furthermore, regular mail offers services which are usually not provided by e-mail, and which are of crucial importance for "official" events.

Certified e-mail tries to provide users with additional guarantees on the content and the delivery of the messages, making e-mail equivalent and in some cases more convenient than the ordinary paper-based mail service. In literature, several distributed protocols for certified e-mail have been proposed, relying on an inline trusted third party (TTP) to ensure the fairness of the protocol. In such protocols, the TTP is actively involved in each message exchange. In this paper we provide a novel inline certified e-mail protocol which satisfies all the most important requirements which have been discussed for certified e-mail. Furthermore, we discuss a prototype implementation of our protocol targeted to the Windows platform.

1 Introduction

The electronic mail service allows users connected to the Internet to exchange messages containing text or multimedia files. The ease of use of e-mail clients as well as the spreading of the Internet and its associated services has determined a large diffusion of the e-mail service. E-mail is more and more used in place of ordinary mail. However, the use of e-mail in official events poses some problems. Indeed the actual e-mail service is based on the Simple Mail Transfer Protocol (SMTP [2]) which offers no guarantees on the delivery and the authenticity of the messages. Compared to the ordinary mail service, the e-mail is much less

Y.G. Desmedt et al. (Eds.): CANS 2005, LNCS 3810, pp. 186–199, 2005.
© Springer-Verlag Berlin Heidelberg 2005

reliable: it gives the sender no evidence of having sent a message as well as no return receipt. Furthermore, whenever an e-mail message is received, there is no assurance on the identity of the originator of the message. Even the transmitted message could be eavesdropped over its path from the origin to the destination, and its content could be manipulated or corrupted by a malicious adversary.

Some e-mail clients (e.g., Microsoft Outlook) provide a Read Receipt request facility. Recipients may receive a request for a response to be sent, but they may decline to send the acknowledgement, or could set a switch to forbid confirmations of such a request. Other e-mail clients may simply ignore the request for a receipt. Indeed, such systems give no guarantee that the sender will receive a receipt when the recipient has displayed the message.

IETF RFC 2298 [1] defines a MIME content-type for message disposition notifications (MDNs). An MDN can be used to notify the sender of a message of any of several conditions that may occur after successful delivery, such as display of the message contents, printing of the message, deletion (without display) of the message, or the recipient's refusal to provide MDNs. However MDNs are not enough to satisfy all the properties usually guaranteed by the regular mail service, because they are easily forgeable.

Exploiting the digital nature of the transaction, it is possible to devise methods and techniques that enhance the capabilities of the message transfer protocol, obtaining the same or even additional guarantees with respect to the paper-based counterpart. One example of such guarantees is the following. A registered mail service allows the sender to prove that she sent *a message* at a specific time to a specific destination. Notice that nothing can be said about the *content* of the message sent. In a digital world, the sender may be able to prove that she sent a *message with a specific content* to a destination.

Certified e-mail protocols basically provide the following property: user Bob receives an e-mail message from user Alice if and only if the latter receives a receipt for this communication, i.e., a proof that the message has been delivered to the recipient. The receipt is such that the recipient cannot deny having received the message. Along with this property, many certified e-mail protocols provide other features like confidentiality of the message, proof of integrity, and so forth. Temporal authentication is, in some cases, e.g., patent submissions, a strict requirement. Enhancing e-mail systems with temporal authentication could simplify such kind of applications by reducing them to the simple operation of sending an e-mail. In Section 2 we describe in more detail the most important properties that have been identified in the literature as being crucial for certified e-mail systems.

Recently a lot of research has been dedicated to the problem of designing certified e-mail protocols. Most of the protocols that have been studied involve a trusted third party (TTP for short) which is delegated by the participants to control the behavior of the parties, assist them during the exchange of messages, and resolve any dispute, if necessary. According to the role played by the TTP, protocols have been classified as *inline* or *optimistic*. In inline protocols [10, 28, 17, 23], the TTP is actively involved in each message exchange: both

the parties send their messages to the TTP, which checks for their integrity and forwards them to the intended receiver. As pointed out in [3], all commercial system providing a certified e-mail service [15, 22, 30] implement protocols in this class. The main reason of this choice is due to the fact that inline protocols guarantee accountability since the TTP is aware of each message exchange. In optimistic protocols [5, 6], the sender and the receiver first try to exchange the message by themselves, without the intervention of the TTP and rely on the TTP only for the cases where a dispute arises (maybe because one of the parties is trying to cheat). Certified e-mail protocols can be seen as a special case of *fair exchange* protocols. For this general case there exist protocols that do not require any trusted third party. Such solutions use the notion of *gradual exchange* [24, 18], i.e., the information is exchanged one bit at a time, or are *probabilistic* [11, 20], i.e., fairness is achieved with a certain probability. However these protocols usually rely on assumptions on the computational power of the parties and suffer from a high communication overhead. Non repudiation protocols with *transparent* TTP have been proposed in [21]. The term transparent refers to the fact that at the end of a protocol run, it is impossible to decide on the intervention of the TTP during the message exchange, by looking only at the produced evidences.

An interesting survey of non repudiation protocols and of the different roles played by the TTP has been provided by Kremer et al. in [19].

In this paper we propose a new inline protocol for certified e-mail. The protocol requires six messages to be exchanged among the parties and satisfies all the requirements usually taken into account in literature. As far as we know, this is the first inline protocol that meets all these requirements. Furthermore, we describe a prototype implementation targeted to the Microsoft Windows platform, based on the development of a software module compatible with one of the most used e-mail client applications.

The rest of the paper is organized as follows. The next section describes the requirements for a certified e-mail protocol. In Section 3 previous proposals are reviewed and compared to our protocol, which is described and analyzed in Section 4. Finally, some conclusions follow the description of the implementation in Section 5.

2 Requirements

Certified e-mail protocols ensure that a participant exchanges a message for a receipt, which the receiver should release at the end of the transaction. Indeed, the aim of such protocols is to provide a method for the secure exchange of messages, which is resistant to possible attempts of cheating by the different participants. Since both the message and the receipt are digital objects, certified e-mail protocols can be seen as instances of fair exchange protocols [6]. Such protocols deal with the fair exchange of objects, i.e., at the end of the exchange, both participants get what they expect or nobody gets any valuable information.

In the following we list the main requirements that a certified e-mail protocol should satisfy.

Fairness: In a *fair* certified e-mail protocol, parties should not be able to interrupt or corrupt the protocol obtaining any advantage from the exchanged messages. At the end of the protocol each party should get the desired information or nobody should get any valuable information: the sender should get both sending and delivery receipts, while the receiver should get the e-mail message;
Sending Receipt: Since certified email protocols are interactive protocols that may involve human interaction, it could be desirable that the sender obtains an evidence of the fact that he *started* the process of sending a certified email. Notice that this receipt may not contain any information generated by the recipient, e.g., it is produced by a third authority.
Non-repudiation of origin: The party which originates the message should not be able to falsely deny having originated it; the receiver should get evidence of the exchange to resolve any dispute;
Non-repudiation of receipt: The recipient of the message should not be able to falsely repudiate having received that message; at the end of the protocol the sender should get evidence of the delivery of the message;
Authenticity: The participants to the protocol should be guaranteed on their reciprocal identities and on the identities of the other entities involved in the message exchange;
Integrity: Parties involved in the exchange of the messages should not be able to alter or corrupt the transmitted messages without being detected;
Confidentiality: Only the sender and the receiver should be able to extract the content of the original e-mail message given the exchanged messages;
Timeliness: The duration of the protocol should be finite, so that the exchange procedure terminates successfully or any party can decide to abort the exchange within a predefined time bound;
Temporal Authentication: The starting time of the exchange should be certified and observable by the participants to the protocols; an arbiter should be able to verify the temporal data attached to messages.

To achieve many of the above properties, many protocols rely on a trusted third party. If the TTP has an active role during the message exchange, such protocols are referred to as *inline protocols*. The drawbacks of such protocols is that the TTP has to be online for the whole duration of the exchange and that the correctness of the protocol is entirely devoted to its behavior: any failure of the TTP could compromise the e-mail exchange. A main advantage of these protocols is that they allow accountability. Indeed, all the commercial systems that provide a certified e-mail service implement a protocol of this kind. On the other hand, optimistic protocols have been introduced by Asokan et al. ([5, 6, 7]), relying on the idea that the TTP takes part in the protocol only in case of failures or to resolve a dispute between the parties. The main drawback of this approach is that this class of protocols does not allow accountability.

3 Related Works

Several researchers proposed protocols for certified e-mail using an online third party during the exchange of the message. Bahreman and Tygar [10] proposed an inline protocol requiring six messages to be sent among the parties. In their protocol the sender sends the e-mail message to the TTP, which returns a proof of mailing. Then, the TTP encrypts the message with a session key and sends it to the recipient, who signs the ciphertext and returns the signature to the TTP. Finally, the TTP sends the receipt to the sender and the session key to the recipient. Our protocol is derived from this one. We notice that this protocol presented in [10] does not preserve the confidentiality from the TTP, since the sender sends the e-mail message to the TTP. Furthermore, there is no temporal authentication.

Deng et al. [17] proposed two inline protocols requiring four messages to be sent among the parties. In particular, the second protocol preserves the confidentiality from the TTP, while the first one does not. Coffey and Saidha [16] proposed a non-repudiation protocol which relies on an external time-stamping authority to state the non-repudiation of origin and destination evidence time. Another non-repudiation protocol requiring four messages have been proposed by Zhang and Shi [26]. In such protocol the TTP manages a database containing the session keys used in a protocol run and publishes at the right time, in a publicly accessible database, the keys needed to allow the deciphering of the exchanged messages. One of the main drawback of this technique is that the TTP cannot delete any element from the database as each key should be recovered by a judge in case of a dispute. For this reason the size of the database grows indefinitely.

Schneier and Riordan [23] proposed an inline protocol using a secure database server. Although they claim this server does not need to be trusted, in practice it is a TTP since it should not be able to collude with the sender. In their protocol the sender encrypts the e-mail message with a session key and sends it to the receiver. Then, the receiver asks the sender to publish the session key on a secure database server at a certain time. This message is signed by the receiver and sent to the sender. Afterwards, the sender submits the session key to the server; then, the receiver gets it and decrypts the e-mail.

Several non-repudiation protocols which have been applied to certified e-mail have been proposed by Zhou and Gollman. In [29] the authors present a protocol that requires five messages. The key idea is that the sender and the receiver exchange the signatures on the encrypted message and then interact with the TTP to recover the key and the non repudiation-proofs. In [27] another protocol is proposed where the e-mail message is transmitted from the sender to the receiver through a sequence of trusted third parties. The role of these parties is to deliver the message, collect the receipt signed from the receiver, and route them back to the sender.

Finally, Abadi, Glew, and Pinkas [3] proposed an inline protocol requiring four messages. The protocol does not require any public-key infrastructure. However, the protocol assumes that the TTP has some public keys and that some other authentication mechanism is provided (such as a shared secret) among the participants.

The following table summarizes the properties guaranteed by each of the above mentioned inline protocols. An empty circle means that the property is not satisfied and a bullet means that the property is satisfied. In this paper we propose an inline certified e-mail protocol (last column of the table) satisfying all nine properties.

Property	[10]	[17]	[16]	[26]	[23]	[29]	[27]	[3]	Ours
Fairness	●	●	●	●	●	●	●	●	●
Sending Receipt	●	○	●	●	○	○	●	●	●
Non Rep. Origin	●	●	●	●	●	●	●	○	●
Non Rep. Receipt.	●	●	●	●	●	●	●	●	●
Authenticity	●	●	●	●	●	●	●	○	●
Integrity	●	●	●	●	●	●	●	●	●
Confidentiality	○	○	●	●	○	○	○	●	●
Timeliness	●	○	○	○	○	●	○	○	●
Temp. Auth.	○	○	●	●	○	○	○	○	●

Fig. 1. Summary of properties

4 The Protocol

In this section we provide a detailed description of the protocol. Recall that the goal of the protocol is to allow a sender S to send an e-mail message to a receiver R, in such a way that the properties discussed in Section 2 are satisfied. Some of these properties derive from the use of cryptographic primitives, others derive directly from the protocol.

4.1 Preliminaries

The scenario we consider consists of a number of users who are willing to exchange e-mail messages using a certified e-mail service. The service provides them some additional guarantees on the delivery of the messages and on the security of the communication. To such purpose, the protocol relies on a Trusted Third Party (TTP) actively involved in each message exchange. The TTP is trusted, in the sense that it does not collude neither with the message sender nor with the receiver. Furthermore it is assumed to be reliable. The protocol assumes that each user has a public key, widely available to the other users and whose authenticity can be verified, and a corresponding private key which is kept secret and known only to him. Currently there are a number of techniques that can be used in order to guarantee the above assumption. The most widely used is the existance a public key infrastructure. We just mention certificateless public key encryption schemes introduced in [4] and ID-based encryption schemes [14, 13, 25] as possible alternatives to PKIs. The public key system defines an encryption transformation and an associated decryption transformation which are used to exchange messages between users in such a way that the *confidentiality* of the messages is guaranteed.

4.2 Cryptographic Primitives

In the following we describe the cryptographic primitives used in the protocol.

- $Sig_A(m)$: denotes the pair (m, σ) where σ is the digital signature of the message m using the private key of user A under any secure signature algorithm;
- $h(m)$: indicates the hash of message m using some collision resistant hash function. A collision resistant hash function maps arbitrary length messages to constant size messages such that it is computationally infeasible to find any two distinct messages hashing to the same value.
- $PK_B(m)$: denotes the encryption of message m using the public key of user B under some public-key encryption algorithm.
- $E_k(m)$: denotes the encryption of message m using the key k under some symmetric encryption algorithm.

4.3 Description of the Protocol

The protocol we propose is derived from Bahreman and Tygar proposal [10]. In order to ensure time related properties to the exchange of messages, we add a timestamping service, which is performed by the TTP. Whenever the TTP receives a request from the sender S, it generates and stores a new transaction associated with the arrival time of the message and the message itself. The transaction is stored for the whole duration of the exchange and can be deleted at the end of the protocol or used as a proof to determine the responsibility of the cheater in case of dispute.

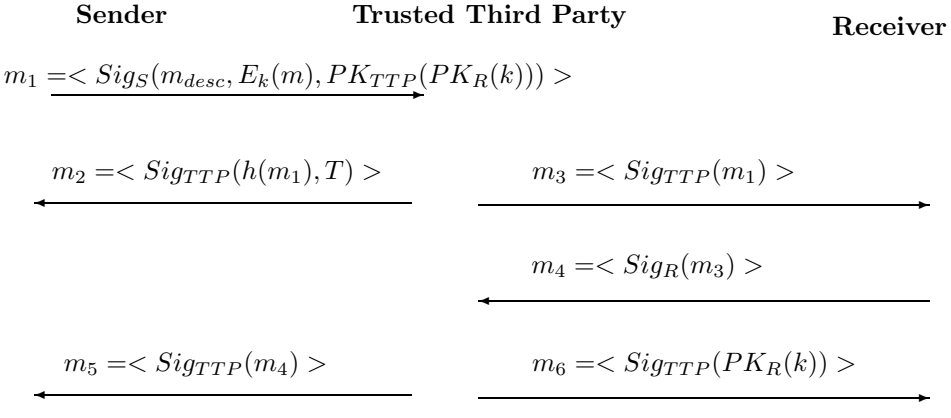

Fig. 2. The protocol

Let us describe in more details the protocol, whose sketch is presented in Figure 2. It is composed of six messages and prescribes the interaction of the sender S with the Receiver R through the TTP.

1. When S wants to send an e-mail message m to R, he chooses a session key k, encrypts the message m using k, encrypts k with the public key of R and then with the public key of the TTP. Afterwards, S adds a brief description m_{desc} of the message m and signs the resulting message, obtaining the message m_1, which is sent to the TTP.
2. At the reception of the message m_1 from S, the TTP generates a timestamp T and concatenates it to the hash of m_1. Afterwards, it signs the resulting message, obtaining the message m_2, which is sent to S. The TTP also signs the message m_1, obtaining the message m_3, which is sent to R.
3. At the reception of the message m_3 from the TTP , R reads the description of the message and decides whether he wants to get the original e-mail message from S. If he does, he signs the message m_3, obtaining the message m_4, which is sent to the TTP. Otherwise, if R is not interested in the message, he can simply ignore the message and abort the transaction.
4. If the TTP receives the message m_4 from R within a time $T < t < T + \delta$, where δ is a predefined time interval, he verifies that m_4 has been obtained by signing the message m_3. In such case, he signs the message m_4, obtaining the message m_5, which is sent to S. Finally, the TTP extracts from m_1 the encryption of the session key k under the public key of R, signs it, and obtains the message m_6, which is sent to R.

4.4 Analysis

In this section we show that our protocol satisfies all the requirements listed in Section 2.

Fairness. If both the sender and the receiver behave as expected and messages are delivered on time, at the end of the protocol each party gets the desired information. Indeed, S gets a sending receipt (that is, message m_2), even if R is not interested in receiving the original e-mail message and aborts the transaction. After receiving the message m_3, containing the encryption of m under the session key k, R has to decide whether he is interested in getting the original e-mail. Only if it confirms to be interested in reading the e-mail, by sending the message m_4 to the TTP, he gets the message m_6, containing the encryption of the session key k under his public key. The protocol ensures that at the same time, the TTP sends the message m_5 to S. This message constitutes a delivery receipt for S, since it contains the signature of the TTP of the message m_4, which is the authenticated confirmation that R was interested in receiving the original e-mail message. Notice that, in case the sender maliciously constructs a message m_1, e.g., by using $E_k(m)$ and $PK_R(k')$ to compose m_1, the receiver will have a proof that the sender cheated during the execution of the protocol.

Sending receipt. The sending receipt consists of message m_2, which contains the signature of the TTP on the hash of m_1. This receipt is sent to S by the TTP before the interaction with R, hence it is independent on the reception of the message from R.

Non-repudiation of origin and receipt. This property is guaranteed since each message sent during the execution of the protocol is signed by its sender.

At the end of the protocol, R gets m_3 and m_6, which represent the non-repudiation tokens of origin. With these tokens, R can prove that S indeed sent the e-mail. On the other end, S gets m_5, which represents the non-repudiation token of receipt. With this token, S can prove that R indeed received the e-mail.

Authenticity, Integrity, Confidentiality. The property derive directly from the authenticity and verifiability of the public key. Integrity come from the collision resistance property of the hash function and on the security of the signature scheme. The protocol also preserves the confidentiality of the e-mail message m, both from the TTP and from an adversary eavesdropping the messages exchanged. Indeed, the TTP never learns the content of the original e-mail message: it receives the encryption of m under the key k, but cannot decrypt it, because k is encrypted with the public key of R. The same holds for any adversary eavesdropping the message exchanges.

Timeliness. The duration of the exchange is finite. Indeed, let τ be the maximum transmission time, i.e., the time needed for a message to reach its destination. Furthermore, let λ be the maximum computation time. It is clear from inspection that, if the sender starts the protocol at time T, the protocol terminates at latest at time $t + 4\tau + 3\lambda$.

Temporal authentication. The TTP guarantees the temporal authentication of the exchange, certifying the starting time T of the transaction (such a timestamp is contained in the message m_2).

5 Implementation

To test the performance and the usability of the proposed protocol, we developed a prototype implementation, targeted to the Windows platform. The implementation relies on the development of applications to manage the messages exchange from both client and server (TTP) side. For the client side, users willing to use the certified e-mail service are requested to install a plug-in, i.e., a piece of software which extends the capability of the usual e-mail client. The plug-in has been designed for Outlook 2000. The extension is conceived in such a way that users are given the option to choose if sending a normal e-mail or using the certified e-mail service.

For the server side, an extension to Exchange Server 2000 has been developed, through the generation of a DLL ActiveX which reacts to the reception of certified e-mail messages and generates the requested messages needed to execute the protocol.

5.1 Previous Implementations

In literature several implementations of certified e-mail systems exist. Generally they can be roughly classified as research projects or commercial systems.

TRICERT [8] is an hybrid scheme based on Postal Agents which are distributed servers acting on the behalf of the TTP. The PA must be online, but they are not able to resolve disputes. The PA have been implemented by daemons included in the Apache web servers, while the client applications provide user interfaces through SSL enabled web servers. The trusted party in practice is a human service operator which has to manage the requests logged into the trusted server. Notice that this scheme is neither optimistic nor inline, so it is not possible to compare it with our proposal.

The protocol proposed by Abadi et al [3] has been implemented by combining the usage of a standard e-mail client with a Java-enabled browser. Certified e-mail messages contain both a plaintext part explaining the content of the message and an HTML part containing a link to an applet with appropriate parameters. To read the message, the receiver must double-click on the link to launch the applet and continue the execution of the protocol. While this kind of solution is attractive, since it does not require the users to install any additional software, it has the disadvantage that messages are not easily manageable, since it is necessary to contact the TTP each time one wants to read, print or reply to the received message.

Several companies offer certified e-mail services usually hiding the technical details related to both the protocol and the used application. Usually the parties communicate through a central trusted web server [9, 15]. Another commercial implementation is ZixMail, which has been developed by the ZixIt Company [30]. The service enables the delivery of documents and e-mail messages in a secure way by encrypting and digitally signing the communications. The ZixMail users can choose between two options for message delivery: ZixMail direct method can be used when both the sender and the receiver are provided with a ZixMail client application, which is available also as Lotus Notes or Microsoft Outlook plug-in; ZixMail.net method is used when the receiver does not have a ZixMail client. In the latter case, the receiver can use an SSL enabled browser to retrieve the message.

5.2 Exchange Server Extension

To implement the server side of the certified e-mail service, we relied on the event model which is provided with the Web Storage System used by Microsoft Exchange Server 2000.

A Web Storage System is a hierarchical folder system which can store all sorts of documents and data types such as e-mail messages, Web pages, multimedia files, and so on. The Web Storage System provides access to events which fire when certain tasks occur within the Web Storage System, for example whenever an item in the store is saved or deleted.

The event paradigm consists of two main elements: the *event sink*, which is the code that Exchange 2000 executes when an event fires in a specified folder; and a *registration element*, which is a hidden item created in the store at the root of the folder, holding all the information necessary to associate the event with its event sink and its properties.

In our case, a new event sink (`eCertifiedMail.dll`) implemented as DLL ActiveX has been created and registered on the server as a new COM+ component. Each time a new message is received by the TTP the event sink registered with the OnSyncSave event is called before the message is stored into the *Inbox* folder, such that the message can be manipulated according to the protocol specification.

5.3 Cmail Plug-In

The plug-in is based on the design of a COM Add-In, which is an ActiveX DLL able to interact with applications coming with the Office 2000 package.

To store the certified e-mail messages in input and output, two new folders have been created in *Inbox* and *Outbox* system folders of Outlook. These folders include other sub-folders which contain the messages generated during the interaction with the TTP. More in detail, in *Outbox*, the *CEMSender* folder contains the subfolder where both sending and return receipts are stored for certified e-mail messages originated from the Sender. The *CEMReceiver* folder contains two subfolders, *Requests* and *Keys*, containing the first and the last messages sent by the TTP and generated during the execution of the protocol whenever an incoming certified e-mail message is received.

The basic cryptographic operations are performed exploiting the cryptographic primitives provided with the Windows operating system, called *CryptoApi*. To this purpose, we used the Visual Basic COM wrappers for the *CryptoAPI* called the *WCCO (Wiley CryptoAPI COM Objects)* [12]. However, to overcome some limitations we developed an extension of the *WCCO* library providing the cryptographic functions needed during the execution of the protocol.

The plug-in is activated whenever the user decides to compose a new message. In this case, a form is displayed on the screen asking the user if she wants to use the usual service or certified e-mail service. If the user chooses to compose a certified e-mail message, a new session key k_1 which is 168 bit long is generated and its hash is calculated and stored as a message-id. Since the *CryptoApi* only allows to use public key encryption schemes to encrypt "short" messages, i.e., session keys and hash values, the part of message containing $PK_{TTP}(PK_R(k))$ could not be implemented as stated. We have substituted this part of the message with $(E_{k_2}(PK_R(k_1))||PK_{TTP}(k_2))$, where k_2 is another randomly generated session key.

To summarize, the final message is created with an appropriate header, holding in some user-defined fields the message type (in this case new certified e-mail message), the subject, the encrypted message $E_{k_1}(m)$, the other information needed by the TTP to continue the exchange, i.e., $E_{k_2}(PK_R(k_1))||PK_{TTP}(k_2)$ and the signature of the sender of all the above components, i.e.:

$$Sig_S(Header||Subject||E_{k_1}(m)||E_{k_2}(PK_R(k_1))||PK_{TTP}(k_2)).$$

The resulting message is sent to the TTP and a copy is stored in the *CEMSender* folder.

Whenever the TTP receives a certified e-mail request from the Sender, after verifying the integrity and the authenticity of the message, it stores the attachments on the disk, calculates and stores a fresh timestamp for the execution of the protocol, and composes two messages m_2 and m_3 for the Sender and the Receiver, respectively.

The Sender verifies the integrity and the authenticity of the message sent by the TTP and stores it in the *Receipt* sub-folder of the *Inbox*.

When the Receiver opens the message sent by the TTP a form containing the Sender and the Subject of the e-mail is displayed, and he is asked to refuse or accept the e-mail. In the first case, the message is deleted from the *Inbox*. In the second case, the attachments are saved on the disk, a verification of the message is performed and a copy is stored in the *Request* folder. Then, a new message (m_4) is composed and sent to the TTP.

On the reception of the acknowledgment from the Receiver, the TTP verifies it and composes two new messages for the Sender and the Receiver.

The Sender verifies and saves the message m_5 in the sub-folder Receipt of the *CEMSender* subfolder.

The Receiver retrieves the session key k_1 to decrypt the message and stores the last message holding the key in the Key sub-folder of the *CEMReceiver* of the *Inbox*.

6 Conclusions

We have presented a new certified e-mail protocol relying on an online trusted third party. Our protocol enhances the basic e-mail system with *all* the most important features discussed up to now in the literature and reported in Figure 1. In particular the properties of confidentiality and temporal authentication should help to overcome the problems that have prevented the use of e-mail for official communications.

The prototype implementation we provide is composed by an extension for Microsoft Exchange 2000 and by a plug-in developed for the Microsoft Outlook e-mail client. Users can continue to use the regular e-mail service and adopt the certified e-mail facility as an additional service. Since our goal was to demonstrate the applicability of the protocol, currently the prototype simply assumes the public key to be known. It is our intention to enhance the basic prototype so that it can interact with a PKI in order to obtain and verify public keys. Finally we are planning to develop other plug-ins for the most used e-mail clients to confirm the practicality of the approach.

References

1. RFC 2298. An extensible message format for message disposition notifications, `http://www.ietf.org/rfc/rfc2298.txt`, March 1998.
2. RFC 2821. Simple mail transfer protocol (smtp), `http://www.ietf.org/rfc/rfc2821.txt`, April 2001.

3. M. Abadi, N. Glew, B. Horne, and B. Pinkas. Certified email with a light online trusted third party: Design and implementation. In *Proceedings of Eleventh International World Wide Web Conference*. ACM Press, New York, US, 2002.
4. S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.
5. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *ACM Conference on Computer and Communications Security*, pages 7–17, 1997.
6. N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, 1998.
7. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In Kaisa Nyberg, editor, *EUROCRYPT 98*, pages 591–606. Springer-Verlag, 1998.
8. G. Ateniese, B. de Medeiros, and M. T. Goodrich. TRICERT: A distributed certified E-mail scheme. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, San Diego, CA, February 2001.
9. http://www.authentica.com.
10. A. Bahreman and J. D. Tygar. Certified electronic mail. In Dan Nesset and Robj Shirey, editors, *Proceedings of the Symposium on Network and Distributed Systems Security*, pages 3–19, San Diego, CA, February 1994. Internet Society.
11. M. Ben-Or, O. Goldreich, S. Micali, and R.L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1), 1990.
12. R. Bondi. *Cryptography for Visual Basic: a programmer's guide to the Microsoft CryptoAPI*. John Wiley and Sons, Inc., New York, NY, USA, 2000.
13. D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In Cramer, editor, *EUROCRYPT 05*, pages 440–456. Springer-Verlag, 2005.
14. D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.
15. http://www.certifiedmail.com.
16. T. Coffey and P. Saidha. Non-repudiation with mandatory proof receipt. *ACM Computer Communications Review*, 26, 1996.
17. R. H. Deng, L. Gong, A. A. Lazar, and W. Wang. Practical protocols for certified electronic mail. *Journal of Network and System Management*, 4(3), 1996.
18. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6), 1985.
19. S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, 2002.
20. O. Markowitch and Y. Roggerman. Probabilistic non-repudiation without ttp. In *Proceedings of Second Conference on Security in Communication Networks*, 1999.
21. O. Markowithc and S. Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In *Procceding of Information Security Conference (ISC 2001)*, volume 2200 of *Lecture Notes in Computer Science*, pages 363–378. Springer, 2001.
22. http://www.readnotify.com.
23. J. Riordan and B. Schneier. A certified E-mail protocol with no trusted third party. In *Proceedings of the 13th Annual Computer Security Applications Conference*, pages 347–352, 1998.

24. T. Tedrick. Fair exchange of secrets. In *Proceedings of Crypto '84*, volume 196 of *Lecture Notes in Computer Science*, pages 434–438, 1985.
25. B. Waters. Efficient identity-based encryption without random oracles. In Cramer, editor, *EUROCRYPT 05*, pages 114–127. Springer-Verlag, 2005.
26. N. Zhang and Q. Shi. Achieving non-repudiation of receipt. *The Computer Journal*, 39(10):844–853, 1996.
27. J. Zhou and D. Gollmann. Certified electronic mail. In *Proceedings of ESORICS '96*, volume 1146 of *Lecture Notes in Computer Science*, pages 160–171, 1996.
28. J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 55–61, Oakland, CA, 1996. IEEE Computer Society Press.
29. J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.
30. Zixmail and zixmail.net. http://www.zixmail.com.