TRICERT: A Distributed Certified E-Mail Scheme

Giuseppe Ateniese Breno de Medeiros Michael T. Goodrich

Department of Computer Science
The Johns Hopkins University
E-mail: {ateniese,breno,goodrich}@cs.jhu.edu

Abstract

In this paper we present protocols for distributed certified e-mail, which use encryption to ensure both confidentiality and fairness. As with other protocols for certified e-mail, ours achieve fairness by placing trust on an external entity, referred to as the Trusted Third Party (TTP). The TTP can become a bottleneck, however, and we explore scenarios that support a distributed TTP, in the context of both off-line and online protocols. With several servers dividing the TTP responsibilities, the level of confidence placed in individual servers can be reduced without compromising the TTP's overall trust.

Keywords: *Secure E-commerce, Fair Exchange, Certified E-mail, Semi-trusted TTP.*

1 Introduction

Considerably valuable data - commercial, medical, educational and scientific - is today stored in electronic format in computer databases. The commercialization of this information, as well as other electronic items of intrinsic value - such as software code, for instance - via internets (or the Internet) is a development of great economical and technological impact, allowing companies to operate with greater efficiency and increase the value of their products, and facilitating faster development of medical and scientific innovations. The fact that this information is intrinsically valuable then has implications as to how it must be transmitted or exchanged. In the real world, when we purchase something, a receipt is issued simultaneously with our receiving the product. Several protocols have been devised which theoretically allow simultaneous exchange of electronic items between two computers. However, most of them demand too much computational power and/or communication bandwidth to be implemented. The lack of simultaneity in electronic transactions creates an issue involving fairness: If the purchaser issues the receipt before obtaining the product he may be denied that product later on, while charged nevertheless for it. Symmetrically, the purchaser may refuse to pay for a product it has received before issuing a receipt, claiming that there is no proof he/she ever purchased such an item. Thus, if two mutually untrusted parties wish to engage in an exchange of an electronic item for its receipt they will need to follow a series of protocol steps that make it impossible - or at least too hard or expensive - to misbehave.

One example of fair exchange service is certified e-mail delivery, in which the recipient gets the mail content if and only if the mail originator receives a proof-of-receipt from the recipient. The proof-of-receipt is generally a signature that can be used to trace the transaction and certifies the mail content. This is different from the traditional certified mail protocol operated by the U.S. Postal Service, where the recipient's signature is only a proof that *"the recipient has indeed received an item from the sender"*. In particular, there is no guarantee that a specific mail content was sent. Only date of transmission and persons involved are certified.

Online protocols employ a trusted third party (TTP) which acts as a delivery channel. Both parties send their items to the TTP which checks for their integrity, ensures the validity and fairness of the exchange, and forwards the items to the intended receivers. In this paper, we present efficient certified e-mail protocols that make use of a trusted third party but in a *optimistic* way, i.e., the TTP is involved only in case of dispute, which is expected to be a rare event.

We explore new trust models by distributing the TTP's role to several entities having different levels of trust. We, finally, describe hybrid protocols that combine the strengths of both optimistic and online approaches and a core system that implements them. Our protocols can be easily extended and employed to exchange generic electronic items.

2 Digital certified e-mail

A certified e-mail service should therefore use cryptographic tools to provide proof that a particular message was delivered between two parties at a certain time. Moreover, we desire certified e-mail schemes that are fast, fair, and simple. We believe that such schemes should minimally provide:

- **Fairness:** No party should be able to interrupt or corrupt the protocol to force an outcome to his/her advantage. In any instance of the protocol, it should terminate with either party having obtained the desired information, or with neither one acquiring anything useful.
- **Monotonicity:** Each exchange of information during the protocol should add validity to the final outcome. That is, the protocol should not require any messages, certificates, or signatures to be revoked to guarantee a proper termination of the protocol. This is important, because if revocation in needed to ensure fairness, then the verification of the validity of the protocol outcome becomes a bottleneck as it requires TTP's active participation.
- **TTP invisibility:** A TTP is *visible* if the end result of an exchange makes it obvious that the TTP participated during the protocol.
- **Non-repudiation of receipt:** The recipient of the message should not be able to deny having received the message if indeed the message was delivered.
- **Non-repudiation of origin:** The sender should not be able to deny having sent the message.
- **Confidentiality:** In case the exchange is deemed confidential, the protocol should not need to disclose the message contents to any other party except for sender and recipient. In particular, other trusted or semi-trusted parties acting as intermediaries should not be able to read the contents of a confidential e-mail.
- **Realistic trust models:** The trust model should be based on realistic assumptions the users are comfortable with. A system that places less trust in outside parties is more likely to be accepted.
- **Efficiency:** The protocol should not involve excessive computational or communication costs. It should let itself to reasonably fast implementations.
- **Timeliness:** Roughly speaking, *timeliness* guarantees that both parties will achieve their desired items in the exchange *within finite time* or that at least one party has the ability to decide to abort the normal operation of the protocol and adopt a scheme for protocol resolution that can be executed in a finite, eventually short, period of time.

We stress the point that a certified e-mail system must assure confidentiality. Most of the protocols for fair exchange of electronic items proposed so far do not provide any level of confidentiality, in the sense that they allow a TTP (which is needed as an arbitrator to ensure fairness) to see the contents of the exchange, at least in the exceptional cases in which there is a dispute and active arbitration by the third party is needed. While our protocol also requires the existence of a trusted arbitrator, the arbitration can be performed without violating the confidentiality of the exchange. Indeed, e-mail messages can be certified without revealing their contents to third parties. Such confidentiality could actually improve the usage of a commercial certified e-mail service, as users might wish that private information not be utilized by third parties for commercial or other purposes. This desire would even be true for a third party that is a major corporation, which people trust to perform transmission, storage, or dispute resolution, but do not trust to keep confidential information private. In fact, such desires are even written into law in many European countries.

In the next section, we review existing protocols and compare them with our own according to the criteria above.

3 Related work

The certified e-mail problem is related to the more generic cryptographic problem of *fair exchange*. The classic solution to this problem (e.g., see [18]) involves the gradual exchange of information between the two parties. Even *et al.* [10] introduced the randomized approach where two parties exchange items one bit at a time. At some point, either party can use a brute-force algorithm to complete the exchange; hence, the scheme gradually converges to one that achieves a probabilistic notion of fairness, albeit with a large number of communication rounds. Likewise, this classic approach implicitly assumes that the two parties have equal computational power, which is unrealistic. Ben-Or *et al.* [5] give an alternate gradual approach. In their scheme, the parties gradually release information that incrementally increases the probability that a fair exchange is valid, with this probability going to 1 after many rounds. Thus, while it reduces the need for equal computational power between the parties, this scheme is still expensive from a communication point of view.

Because of the high communication costs of gradual exchange schemes, more recent work has focused on the use of TTPs to make fair exchange more efficient. Asokan *et al.* [1, 2] describe very efficient *optimistic* protocols for the fair exchange problem. When applied to certified e-mail, their (asynchronous) scheme [2] results in five messages being sent when there is no dispute. The protocol in [2], however, is not strictly monotonic: In order to achieve complete fairness, some messages might have to be revoked by the TTP. The protocol

in [1] is expensive as it makes use of verifiable escrow protocols implemented via a *cut and choose* method. However, it provides timely termination assuming only resilient channels.

Notice that, certified e-mail is *asymmetric*, i.e., the recipient can send back a receipt only after he has received the message in some form (possibly even encrypted), whereas the fair exchange involves simultaneous exchange of two items. This difference may be crucial and can be exploited in order to find more efficient solutions than those for fair exchange. Many existing approaches address specifically the certified e-mail problem rather than the more generic fair exchange problem.

The Internet service `www.certifiedmail.com` provides certified e-mail using its server as a TTP. It is an *online* service where the TTP acts as transmission medium for both the message and its receipt. The scheme is very simple: The sender sends the message to the TTP which informs the recipient that a message for him has arrived. The recipient authenticates himself with a password and, then, reads the message. Finally, the TTP sends a signed receipt to the sender that the recipient has indeed received and read the message. Although simple and efficient, the scheme requires an online TTP for every transaction, it does not provide confidentiality, and it does not actually give the sender a receipt signed by the recipient, as the receipt is signed by the TTP only.

There are several published protocols specialized for certified e-mail, as well. Zhou and Gollmann [21] give an online certified e-mail protocol, which involves transferring the message from the sender though a series of TTPs, delivering the message, collecting the receipt from the recipient and routing it back to the message originator. They also provide a version of their protocol where the recipient signs a hash of the message before he can read it. The trusted party in their protocol consists of replicated servers. This means that each server must be trusted in order for the protocol to work properly. One single compromised server would invalidate the entire scheme.

Bahreman and and Tygar [3] present an elegant online strategy that uses six messages. In their scheme, the sender sends the message to the TTP, which returns a proof of mailing. The TTP, then, encrypts the message and sends it to the intended recipient, who signs the ciphertext and returns the signature to the TTP. Finally, the TTP sends the receipt to the sender, and the deciphering key to the recipient. Notice that, the receipt is signed by the recipient, however this system assumes that the online TTP is fully trusted. Moreover, confidentiality from the TTP is not discussed.

Deng *et al.* [7] also provide an online protocol for certified e-mail. Their scheme requires only four messages to be sent, which is obviously optimal for online protocols requiring a proof-of-receipt from the recipient. Even so, their scheme does not achieve confidentiality from the TTP.

Schneier and Riordan [17] present two protocols, one online and the other optimistic, where the TTP is a secure archiving message database. (The authors present the TTP as a public publishing location, which might be implemented as a secure database server.) In their online protocol, the sender, Alice, sends an encrypted message to the recipient, Bob. Bob, then, replies with a dated, signed request for the decryption key. Then, the sender submits the key to the TTP, from where Bob can retrieve it. Alice's proof-of-receipt consists of the signed request from Bob and the database record kept with the TTP. In the optimistic version of their protocol, Alice sends the decryption key directly to Bob, and Bob sends Alice a receipt of the key. If Bob doesn't reply, then the protocol reverts to the online version. Their optimistic approach implies a visible TTP, as the form of the receipt from Bob is different depending on whether the protocol worked optimistically or not. Moreover, the TTP must be directly involved in any secondary adjudication as it must provide, in the case involving dispute resolution, an additional signed proof-of-mailing with each query or deposit.

3.1 Prior work on Degree of Trust

Franklin and Reiter [11] introduce the notion of a semi-trusted third party for the fair exchange problem. Their protocol is online as it requires the TTP to be involved in any transaction. The TTP can sometimes fail or misbehave but it cannot conspire with either of the parties involved in the exchange. Their model is actually more restrictive, it is assumed that at most one party misbehaves. If the sender cheats, for instance, then the recipient and the trusted third party must be both honest. This also implies that if the TTP misbehaves then, by definition, the other two parties are honest and, in principle, they could simply exchange their items by themselves.

4 Our results

In this paper we propose a scheme for distributed certified e-mail, which we call TRICERT. Our main motivation is to find a model for certified e-mail that would allow for efficient and easy-to-implement schemes. There are many efficient protocols for certified e-mail but very few are practical. Efficiency is generally interpreted in theoretical terms but very rarely it is considered from a practical point of view. In fact, the only protocols for certified e-mail implemented and operational so far are

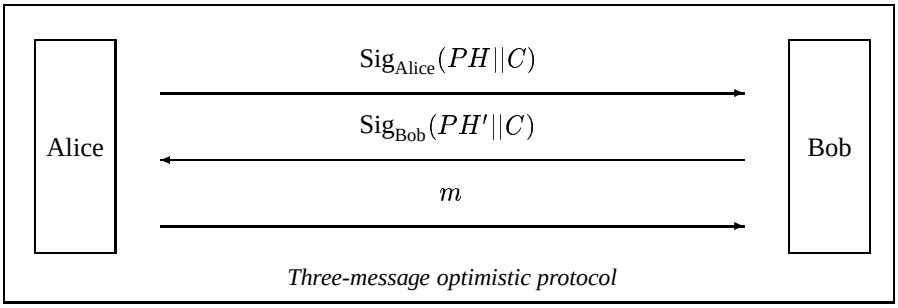

Figure 1: Sequence of messages in the optimistic exchange.

those online, i.e., that require a TTP to be involved for each exchange.

TRICERT is a hybrid protocol that combines the strengths and overcome the disadvantages of both optimistic and online approaches.

The TRICERT scheme *scales*. We introduce the notion of postal agents (PAs), which are distributed servers acting on behalf of the TTP, with each PA requiring minimal trust in itself. The PAs are online but they do not resolve disputes, which are still handled by the TTP. The protocol is monotonic, in that each party cannot revoke a message after it has been sent (like physical certified mail) and makes use of "off-the-shelf" cryptographic technology, such as digital signatures and public-key cryptography. Additionally, the protocol provides TTP's invisibility, and achieves confidentiality from both the TTP and the PAs, which are able to verify the validity of a proof-of-receipt and proof-of-origin without knowing the e-mail content.

We extended the model of Franklin and Reiter [11] as the PAs are semi-trusted, in the sense that they can fail or misbehave, but in addition they can conspire with either of the parties involved in the exchange.

Before we describe TRICERT, we show, in Section 5, that three messages are sufficient to perform optimistic certified e-mail, which is actually one message below the lower-bound for the online case. This three-message optimistic protocol is hardly novel and does not constitute the contribution of this paper. In fact, we later discovered that the general idea behind it is described in a US patent by Micali [16]. Nevertheless, we will extend it and show how to make it more practical in Sections 5 and 6. In particular, we add timestamps and lifetime parameters and show how to distribute the trust of the TTP via threshold cryptographic techniques.

5 A simple optimistic protocol

Before we discuss issues of distributing the trusted party, we introduce a simple optimistic protocol that we will use as sub-protocol in our implementation (see Section 6). In the optimistic approach, the protocol terminates successfully without intervention of the trusted party if sender and recipient both act honestly. Only in case of dispute, the TTP is involved. The general idea is as follows ([16]): The initiator, Alice, first encrypts a message m with the public-key of the recipient, Bob. The result, $\mathrm{P}_{\mathrm{Bob}}(m)$, is further encrypted under TTP's public-key, achieving $Z = \mathrm{P}_{\mathrm{TTP}}(\mathrm{P}_{\mathrm{Bob}}(m))$, and finally sent to Bob. Bob, then, issues a receipt by sending Alice his signature on Z. Upon verifying the receipt, Alice sends Bob the message m. If Alice does not reply, Bob sends Z and his signature on it to the TTP which will then recover $\mathrm{P}_{\mathrm{Bob}}(m)$ and give it to Bob, while forwarding Bob's receipt to Alice. Since the message was first encrypted with Bob's public-key, the confidentiality of the transaction is guaranteed even in this special case.

This protocol is extremely simple. However, we need to slightly modify it in order to achieve timely termination. In particular, a time limit must be incorporated in the protocol otherwise Bob might never reply or might decide to resolve the protocol with the TTP after a certain amount of time that may be unacceptable to Alice. In the next section, we describe the actual protocol we have included into our system implementation.

5.1 Protocol Description

Alice wishes to send a message m to Bob, and wants a signed receipt back. Alice produces an identification token for herself, containing her name and e-mail address for responses, and other identifying information (such as a public-key certificate). We refer to this by Id_{A}. The generation of this token involves no secrets and can be done by any entity from publicly available information about Alice. In fact, Alice also generates (or retrieves) a similar token for Bob (Id_{B}) and for the TTP ($\mathrm{Id}_{\mathrm{TTP}}$).

The identification tokens will be combined with other parameters, such as a timestamp, a nonce n_A (a random number), and a time limit in a protocol header (PH). Notice that, in this protocol, both timestamps and nonces are needed to prevent replay attacks. The

header also should contain other pertinent information about the protocol, such as the encryption, authentication and signature algorithms used. Thus, we have:

$$PH = \{\mathrm{Id_A}||\mathrm{Id_B}||\mathrm{Id_{TTP}}||\text{protocol descriptors}\},$$

where $||$ denotes the concatenation operation. Alice, then, encrypts m with Bob's public-key and concatenates the result with PH, which is subsequently encrypted under TTP's public-key. The resulting ciphertext is:

$$C = \mathrm{P_{TTP}}(PH||\mathrm{P_{Bob}}(m)).$$

Alice concatenates the above with the protocol header, signs it and sends the resulting signature $\mathrm{Sig_{Alice}}(PH||C)$ to Bob[1].

Bob receives the message, and, from PH, he gets relevant information to properly generate a receipt. Bob can discard the message or he may decide to read the content, which implies a receipt must be sent to Alice. The receipt is a signature of Bob, $\mathrm{Sig_{Bob}}(PH'||C)$, stating that he has indeed received a message encrypted as specified in PH. The new protocol header PH' contains a new timestamp and the specifications of the signature algorithm. It also includes the old PH and indicates that the signed message is indeed a receipt. Upon receiving Bob's receipt, Alice releases the message m.

The sequence of protocol messages can be seen in figure 1.

The only place where the protocol can be interrupted with an unfair outcome is after the transmission of the second message, when Alice has Bob's signature but Bob cannot yet read the message. If indeed Alice does not send Bob the third message, Bob contacts the TTP, forwarding the contents of the messages in the first two steps. The TTP decrypts C, checks the protocol headers, and then verifies Bob's receipt. If all is correct, it gives Bob the message $\mathrm{P_{Bob}}(m)$ and forward the receipt to Alice (in case Bob didn't send the second message before complaining).

Notice that, Bob is signing encrypted information which constitutes *a statement to the fact that he received the message.* This is made explicit in the receipt by the concatenation of the protocol header PH' with the encrypted message. Since Bob can take steps to ensure recovery of the message contents, he cannot repudiate his signed receipt on the sole basis that the message received was encrypted and unintelligible. The verification of the receipt can be done by encrypting twice the message m in order to compute C and then checking Bob's signature via public verification algorithms specified in PH'. Alice must also provide the signed message of the first step of the protocol.

In the actual implementation, the message in the third step is concatenated with another protocol header in order to allow the recipient to properly link this protocol step with the two previous ones. Notice that, if confidentiality was not desired, the encryption with Bob's public-key could be avoided without compromising the other guarantees of the protocol.

[1]The notation $\mathrm{Sig_{Alice}}(\cdot)$ implies that the signed plaintext is also available, either because the signature scheme allows for message recovery or because the plaintext is attached to the signature.

6 TRICERT

As already mentioned, the model introduced by Franklin and Reiter [11] for online fair exchange is quite restrictive. Their protocol cannot be easily adapted for certified e-mail, mainly for two reasons: It is assumed that at most one party fails or misbehaves, and that each party knows the one-way hash value of the item that is expected in the exchange. These are unrealistic assumptions in our environment. Nevertheless, the TTP need not be fully trusted and this is appealing for protocols on large networks. The costs of realizing and maintaining a semi-trusted server are much lower than those incurred for a fully-trusted third party. However, the TTP cannot conspire with either of the communicating parties.

We propose a hybrid scheme that achieves the benefits of the optimistic and online protocols. In our model, we consider a highly-secure and fully-trusted server (TTP) and several low-cost semi-trusted servers, which we refer to as Agents. In a fair exchange scheme, the Agents are directly involved in the exchange but they can misbehave or simply crash, in which case the TTP is invoked in order to handle this exceptional case. Our protocol distributes responsibilities so that the TTP need not be highly available, thus lowering the communication demand on it. The Agents are semi-trusted servers acting as intermediary between the two parties involved in the exchange. This increases the availability of the entire system at a lower cost. Most importantly, in our model, the Agents are allowed to conspire with either of the main parties.

In the next Section, we describe our hybrid protocol specialized for certified e-mail. The Agent server involved is called Postal Agent (PA) and is initially chosen by the message originator. Because of this, we simplify our model and make it more practical by assuming that PA will not conspire with the recipient of the message.

6.1 Protocol Description

The TRICERT scheme starts with Alice recruiting the postal agent (PA) to intermediate the interchange in her

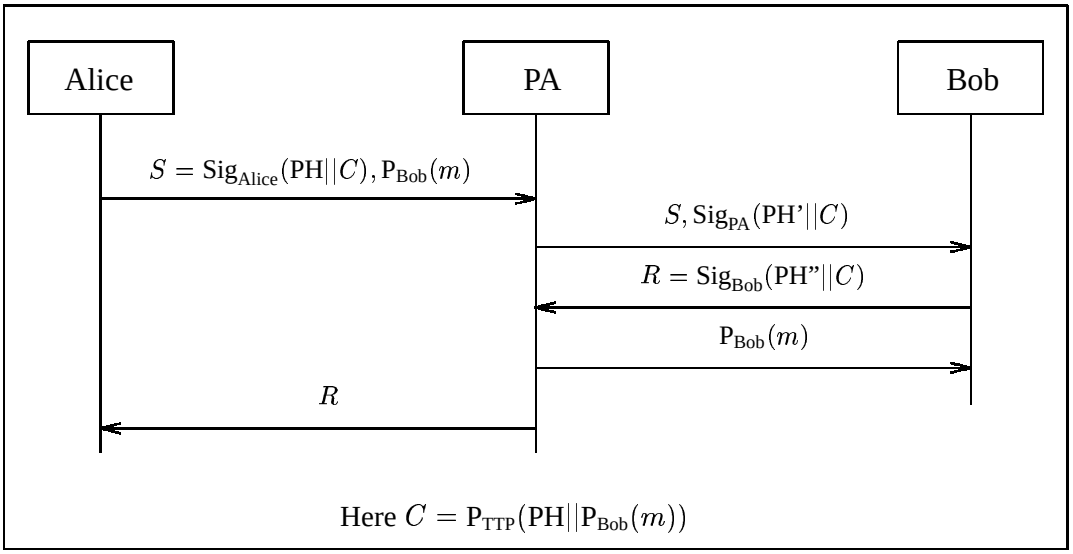

Figure 2: TRICERT scheme – protocol 1

behalf. She gives PA the message m encrypted with Bob's public-key ($\text{P}_{\text{Bob}}(m)$). Then the protocol proceeds with an optimistic exchange between PA and Bob. At the end, PA forwards Bob's receipt to Alice. We also assume that **the communication is performed through private and authenticated channels.**

We have two versions of our protocol. The first version requires five messages to be completed and the second one only four, which is optimal.

Protocol 1. The five-message version works as follows: Alice encrypts the message m first with Bob's public-key, and concatenates the protocol header PH to the ciphertext. She then encrypts the result under TTP's public-key and signs it. The signature is sent to PA along with $\text{P}_{\text{Bob}}(m)$. Optionally, Alice could ask PA to provide her with a proof-of-mailing (a receipt from PA) in reply to her first message. Next PA and Bob perform an optimistic exchange. Specifically, PA sends Bob the request from Alice along with its own commitment to the transaction in form of a signature. Bob checks the signatures and sends the receipt to PA which replies with the encrypted message $\text{P}_{\text{Bob}}(m)$ while forwarding the receipt to Alice. The message flow diagram can be seen in figure 2.

PA can fail or conspire with Alice. Bob can complain with the TTP if he does not receive the last message from PA, in which case, he forwards to the TTP the content of the first message received from PA and his receipt. As in the optimistic protocol, the TTP performs the necessary checks, sends $\text{P}_{\text{Bob}}(m)$ to Bob and, finally, forwards Bob's receipt to Alice. The signature of Alice, S, constitutes the proof-of-origin. Moreover, each protocol header, such as PH, must include the identities of all parties involved. In particular, it must include the identities of Alice, PA and Bob, as well as the TTP's identity in case of multiple TTPs. In addition, PH must be included in the encryption under TTP's public-key. All is done to prevent subtle replay attacks. For instance, Bob could claim that the encrypted message m had been sent to him by a colluding partner. The TTP would then decrypt the message for Bob and forward Bob's receipt to the cheater, who would conveniently (for Bob) dispose of it. As before, a time limit must be included in the protocol headers, which implies that Bob cannot recover the message after that specified time. Since a proof-of-origin is useless without the corresponding message body, Alice's liability immediately ends after the time limit if Bob has not recovered the message (and provided Alice with the receipt).

Protocol 2. The second version of TRICERT is very similar to Protocol 1 but it requires only four messages which is optimal for online protocols. Our protocol improves over existing solutions as the postal agent can misbehave or fail and, in particular, conspire with the message originator. This is achieved as follows: Alice recruits a postal agent PA to act as intermediary. She sends the signature S in Protocol 1 directly to Bob along with $\text{P}_{\text{Bob}}(m)$ but encrypted under PA's public-key. Bob checks the signature and generates a receipt. Notice that, Bob cannot read the message m since it has been encrypted for the postal agent. Alice's message is then forwarded to PA by Bob, along with the receipt.

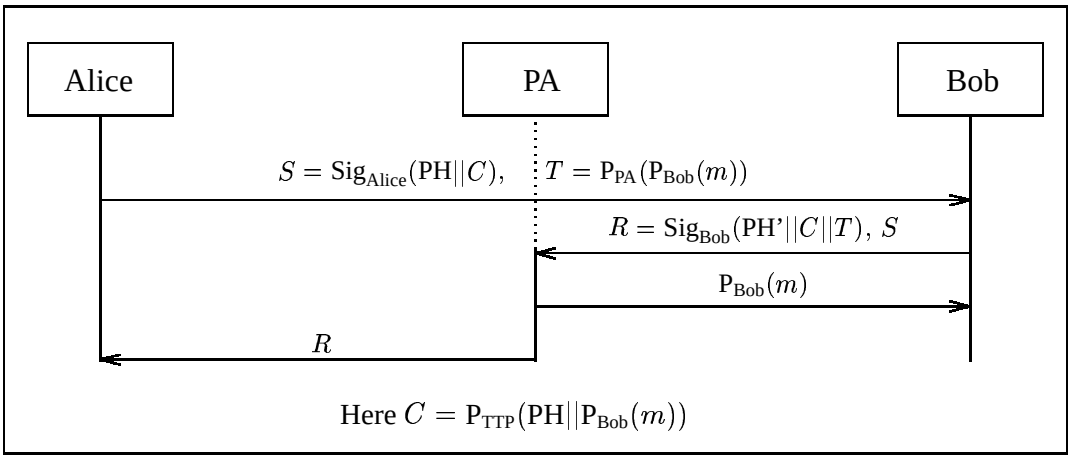

Figure 3: TRICERT scheme – protocol 2

If the receipt is valid, it is sent to Alice by PA which also forwards $\text{P}_{\text{Bob}}(m)$ to Bob. In case of dispute, Bob contacts the TTP as usual. For details, see figure 3.

An important point to notice is that in this version of the protocol Bob should sign both C and T in the second protocol message. Observe that in the first version with five messages, Bob only needs to complain to the TTP if the PA is not forthcoming, an unlikely event; in particular, in that setting if the PA is honest it is not possible for Alice to cheat, as the contents of her message can be verified for consistency by the PA. However in this four message version Alice sends Bob the contents directly, and Bob cannot verify that C and T are linked in any way. Thus if Bob signed only C he would have to rely on the TTP to solve any further disputes, as he does not trust the PA to discard his signature after Alice's dishonesty has been verified. The entire role of the PA would thus be a redundant one. On the other hand, by signing both C and T, Bob safeguards himself against any dishonesty on Alice's part. The verification algorithm is modified to compute both C and T from the private message $\text{P}_{\text{Bob}}(m)$ and to check their consistency as well as Bob's signature on them; and thus Bob will have to resort to the TTP only if the PA misbehaves, exactly as in the first protocol version.

Although this version of TRICERT is more efficient, we decided to implement the five-message protocol. There are reasons for this, strictly related to practical concerns. First, we believe Bob should have a signature from PA before issuing the receipt. This signature constitutes a commitment of PA to the transaction and helps Bob collecting evidences that can be useful in case of dispute. Second, PA may not be willing to act on Alice's behalf at some point. For instance, PA may charge Alice for its service but Alice may refuse to pay. If this is the case, then Alice and PA should first negotiate payment terms and then involve Bob in the exchange, as it may happen in Protocol 1. In Protocol 2, PA may decide not to terminate the protocol, after Bob generated and forwarded the receipt, because of issues with Alice, requiring Bob to complain with the TTP.

Remark 1 Our trust model assumes that Alice, the sender, trusts PA which cannot conspire with Bob by providing him the message without collecting the receipt. We believe this is a plausible assumption since Alice initiates the transaction, freely choosing PA. Within a business model, a contract can be set in which the agent agrees to provide its services to Alice. Bob, on the other hand, while trusting the TTP (as do all parties), does not need to place trust in PA chosen by Alice.

An extension of the original TRICERT schemes is possible, in order to eliminate Alice's need to trust the postal agent. Alice can select several agents and send each the signature S along with a distinct share of the ciphertext $\text{P}_{\text{Bob}}(m)$. Each postal agent would then transfer S and its own commitments to Bob in exchange for the receipt. If the receipt is valid, each agent would send its own share to Bob. Bob can retrieve the ciphertext $\text{P}_{\text{Bob}}(m)$ by pooling together all the shares. This is done via simple secret sharing schemes. If Bob does not receive all the shares or the message is not the one expected, he can complain with the TTP. Bob would still be protected against any of the agents cheating, while Alice would have the guarantee that Bob could not retrieve anything useful unless *all* the agents she hired conspire with Bob.

The shares can be made by xoring the ciphertext with random numbers with the same bitlength. For three agents, Alice would generate two random numbers r_1 and r_2. Then, she would send to the first agent: $S =$

$\text{Sig}_{\text{Alice}}(\text{PH}||C), r_0 = \text{P}_{\text{Bob}}(m) \oplus r_1 \oplus r_2$. The second agent would receive S and r_1 and the third one, S and r_2. If the receipt is valid, Bob receives the shares r_0, r_1, r_2, and then computes the ciphertext $\text{P}_{\text{Bob}}(m) = r_0 \oplus r_1 \oplus r_2$. It is enough that at least one postal agent is honest in order to protect Alice from colluding attacks.

Remark 2 During the receipt verification process, Alice provides the message m which is then encrypted by the verifier twice to achieve the value $C = \text{P}_{\text{TTP}}(\text{PH}||\text{P}_{\text{Bob}}(m))$. Once C is computed, the verifier checks the signature of Bob that constitutes the proof-of-receipt. This verification is also performed by Bob when he receives $\text{P}_{\text{Bob}}(m)$ from PA in order to check that the message he is reading is the same contained in the receipt R. If the message is different, then Bob contacts the TTP to solve the dispute. This implies that the public encryption algorithms $\text{P}_{\text{B}}(\cdot), \text{P}_{\text{TTP}}(\cdot)$ should be deterministic. If they are randomized, then Alice must also provide the random parameters used during the encryption phase.

We adopted a different, more practical, technique. We employed a Message Authentication Code (MAC) to construct a heuristically secure encryption scheme. A practical construction for a MAC function is described in Bellaire *et al.* (see [4]), called HMAC. Then, one would encrypt m as follows:

$$\text{E}_k\left[m||\text{HMAC}_\ell(m)\right],$$

where E_k is a symmetric encryption algorithm, such as DES in CBC mode, and k and ℓ are random session keys. The keys k and ℓ can be encrypted using public-key cryptography, for instance:

$$\text{P}_{\text{Bob}}(k||\ell),$$

where $\text{P}_{\text{Bob}}(\cdot)$ is deterministic, such as plain-RSA. Hence, Alice reveals the random session keys to the verifier during the verification process. Notice that, this encryption method provides also protection against the adaptive chosen ciphertext attack, although this protection is only heuristic.

Remark 3 Our protocols reduce the demand on the fully-trusted party, which needs only to be involved in case of disputes. This situation can be even improved by using threshold cryptosystems [8, 9] instead of traditional public-key cryptography. The idea is to have n TTPs instead of a single one and encrypt messages such that only $t \leq n$ or more TTPs can decrypt them. In a threshold cryptosystem, the secret key is shared among the participants using a t-out-of-n secret sharing scheme. Once the message is encrypted, each participant takes as input the ciphertext and his share and returns as output the original plaintext. If at least t participants follow the decryption protocol, then the original message is recovered successfully.

TRICERT can be easily modified to support multiple TTPs by just selecting the encryption function $P_{TTP}(\cdot)$ as a threshold cryptosystem. This applies also to the optimistic protocol in Section 5.

6.2 Motivation

Compared with online protocols, TRICERT is clearly preferable as it requires four messages, which is optimal for online schemes, and it scales better as it makes use of low-cost semi-trusted third party, the postal agents, that can misbehave or fail.

While simple and elegant, the protocol described in Section 5.1, based on [16], has some disadvantages. It places a too large burden on Bob. We believe Bob is a *passive* participant that receives messages from, perhaps unknown, untrusted senders and shouldn't be so heavily responsible of handling disputes by interacting with the TTP. Once Bob sends his receipt, there is no guarantee that he will receive the message from Alice. So, Bob has to wait for some unspecified amount of time and still must contact the TTP before the time limit. Bob does this for any message he receives; a very unpleasant situation that has not counterpart in the real world. Moreover, the communication channel between Bob and the TTP must be *reliable*, i.e., always operational and without delays, otherwise fairness may not be ensured for Bob as the time limit could expire before he can reach the TTP.

The protocol in [1] assumes only *resilient* channels. A resilient channel will eventually deliver a message sent through it. The time lapse incurred at the conclusion of the delivery process may be arbitrarily long, yet finite. However, the protocol in [1] requires the TTP to keep state and it is quite expensive compared to other solutions.

Our approach mitigates the above issues without requiring the TTP to keep state (the TTP does not store any value). We still require a reliable channel between Bob and the TTP, however the number of disputes is drastically reduced since we expect the postal agent to act honestly more often than a totally untrusted sender. Our protocols may increase Bob's willingness of participating by providing his signature; Bob will only be signing receipts for requests originating from certified agents, rather than from unknown senders. Bob has a further incentive: the duration of his interchange with PA is likely short (in terms of seconds, in our implementation). This because PA is an online server always available whereas Alice, the sender, may not reply promptly, putting Bob at risk.

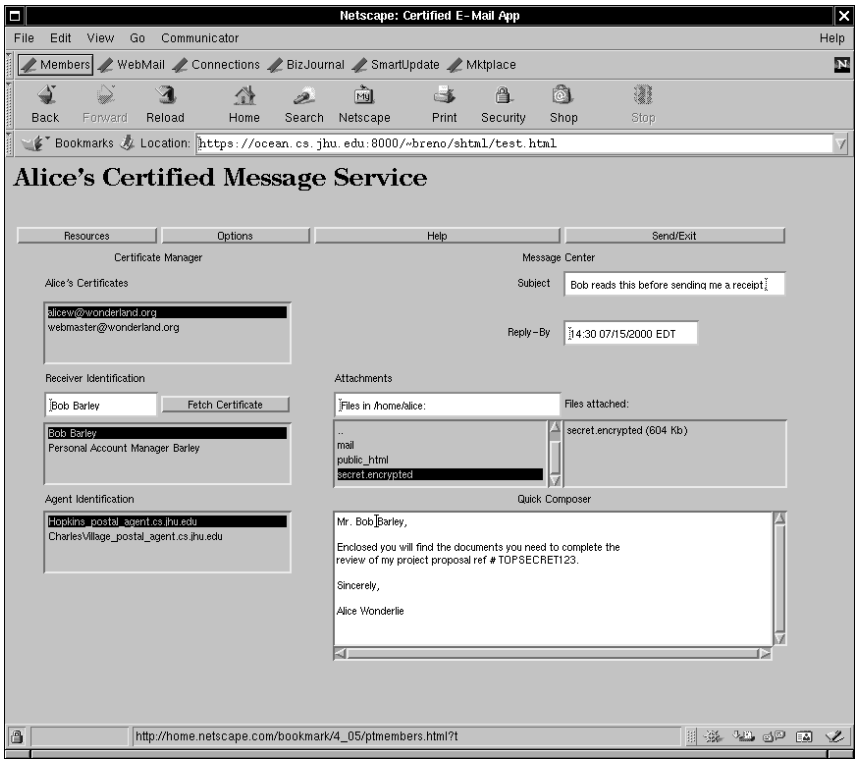

Figure 4: The main GUI frame

7 System implementation

The first implementation decision was which platform to use. Efficiency was only one of the concerns to be addressed. As in the case of online systems, we had to develop both the agent (server) application and the user interface (client application). We wanted our system highly usable and able to support several authentication methods (e.g., PGP-type or X.509 certificates). An important restriction on the platform type is that it should be able to incorporate existing, freely available cryptographic libraries. We employed OpenSSL to provide SSL capabilities (`http://www.openssl.org/`) and a modified Gnu GPG library (`http://www.gnupg.org`) to sign and encrypt messages.

The client application provides a user interface through a SSL-enabled web server. The postal agent servers are currently implemented on two Linux machines running the Apache web server (`http://www.apache.org/`) with the module *mod_ssl* enabled which allows for SSL secure connections (`http://www.mod_ssl.org/`). Daemons running in these computers performed all the agents' transactions automatically. We experimented with Java servlets for the daemons instead of CGI/bin since the web server can satisfy a client request through a single process allowing to handle more transactions simultaneously and more robustly.

The user interface should ideally be a standalone application, with capabilities of web browsing (including SSL connections). In our case, however, we borrowed the same web servers running the postal agents.

The trusted party is clearly the security critical server. We did not run a daemon on this machine. Instead, we designated ourselves service operators. The requests are logged into the trusted server, and the operator immediately prompt for assistance. The trusted server itself is a machine dedicated to this service during our testing phase (a 800MHz pentium III, 256 MB RAM Linux box) that is protected by a firewall and unavailable for remote login.

The client GUI is extremely simple as shown in the screenshot. After pointing it to the secure site, Alice, the sender, is prompted for an ID and password. The leftmost top button, labeled "Resources", if pressed will pop up a window from where Alice can specify the files containing information such as certificates and keys, and also enter bookmarks to web directories from where user certificates/public-keys can be downloaded. Alice's own certificates are displayed in the top left list window. Similarly, the certificates and Ids of the postal agents are shown in the lower left list window. Alice can specify the name of the recipient in the field labeled "Receiver

Identification", and, by pressing the "Fetch certificate" button, download the corresponding certificate which is displayed in the middle left list window. While we expect that most messages will consist only of attached files, a simple text composer is also included for convenience at the bottom right. By pressing the "Send/Exit" button, Alice sends the encrypted request to the postal agent.

The receiver, Bob, will receive a secure URL to point to. Then using a SSL-enabled browser, he provides the receipt and receives back the message promptly.

We found that the system performed satisfactory under the simulating conditions. We plan to extend it by implementing the version with multiple TTPs.

8 Conclusion

In this paper, we presented practical protocols for certified e-mail. We introduced a hybrid approach which combines the strengths of both online and optimistic approaches and allows for effective scalability by distributing responsibilities to low-cost semi-trusted servers. Our protocols are simple and efficient. They articulate realistic and flexible trust models that could be employed to create attractive, usable certified e-mail systems.

9 Acknowledgements

The authors gratefully acknowledge the comments of Michael Steiner, which greatly improved a previous version of this paper. The first author would like to thank Silvio Micali for illuminating discussions on certified e-mail protocols.

References

[1] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. Technical Report RZ 2973, IBM Research, 1997.

[2] 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. IEEE Computer Society Press, May 1998.

[3] A. Bahreman and J. D. Tygar. Certified electronic mail. In *Proceedings of Symposium on Network and Distributed Systems Security*, pages 3–19. Internet Society, Feb. 1994.

[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. pages 1–15.

[5] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, IT-36(1):40–46, 1990.

[6] M. Blum. Coin flipping by telephone – a protocol for solving impossible problems. In *Digest of papers from Compcon Spring 1982*, pages 22–25, 133–137. Feb. 1982.

[7] R. H. Deng, L. Gong, A. A. Lazar, and W. Wang. Practical protocols for certified electronic e-mail. *Journal of Network and Systems Management*, (4(3)):279–297, 1996.

[8] Y. Desmedt. Society and group oriented cryptography: A new concept. In *Advances in Cryptology – CRYPTO'87*, pages 120–127, 1987.

[9] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology – CRYPTO'89*, pages 307–315, 1989.

[10] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Comm. ACM*, 28(6):637–647, 1985.

[11] M. Franklin and M. Reiter. Fair exchange with a semi-trusted third party. In *Proc. ACM Conference on Computer and Communications Security*, 1997.

[12] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Proceedings of Eurocrypt'99*, 1999.

[13] O. Goldreich. A protocol for sending certified mail. Technical report, Computer Science Department, Technion, Haifa, Israel, 1982.

[14] S. Ketchpel and H. García-Molina. Making trust explicit in distributed commerce transactions. In *Proceedings of the International Conference on Distributed Computing Systems*, 1996.

[15] M. Luby, S. Micali, and C. Rackoff. How to simultaneously exchange a secret bit by flipping a symmetrically biased coin. In *Proceedings of the 25^{th} IEEE Symposium on Foundations of Computer Science*, pages 11–21, 1984.

[16] S. Micali. Simultaneous electronic transactions. Technical Report 566420, http://www.delphion.com/cgi-bin/viewpat.cmd/US566420__, 1997.

[17] J. Riordan and B. Schneier. A certified e-mail protocol. *13th Annual Computer Security Applications Conference*, pages 100–106, Dec. 1998.

[18] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., 1996.

[19] B. Schneier and C. Hall. An improved e-mail security protocol. In *13th Annual Computer Security Applications Conference*, pages 232–238. ACM Press, Dec. 1997.

[20] T. Tedrick. Fair exchange of secrets. In G. R. Blakley and D. C. Chaum, editors, *Proceedings of Crypto'84*, pages 434–438. Springer, 1985. Lecture Notes in Computer Science No. 196.

[21] J. Zhou and D. Gollmann. Certified electronic mail. In *Computer Security – ESORICS'96 Proceedings*, pages 55–61. Springer Verlag, 1996.