J. A. Onieva, J. Zhou, and J. Lopez, "Enhancing Certified Email Service for Timeliness and Multicast", *Fourth International Network Conference*, pp. 327-335, 2004.
NICS Lab. Publications: https://www.nics.uma.es/publications

Enhancing Certified Email Service for Timeliness and Multicasting

Jose Antonio Onieva[1], Jianying Zhou[2] and Javier Lopez[1]

[1]Computer Science Department, E.T.S. Ingenieria Informatica
University of Malaga, 29071 - Malaga, Spain
email:{onieva,jlm}@lcc.uma.es
[2]Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
email:jyzhou@i2r.a-star.edu.sg

Abstract

Certified email is a value-added service of ordinary email, in which a sender wants to obtain a receipt from a recipient. Fair exchange protocols are a key component for certified email service to ensure fairness, i.e., the items held by two parties are exchanged without one party obtaining an advantage. We can find in the literature simple and fast optimistic protocols for fair electronic exchange and, more specifically, for *certified electronic mail* (CEM) and *electronic contract signing* (ECS). We have observed that some aspects of those protocols could be substantially improved. This paper presents two major contributions. Firstly, we provide a solution that allows both parties to end the protocol timely in an asynchronous way. Then, we extend the certified email service to the multicast scenario.

Keywords

Certified Email, Fair Exchange, Multi-party Protocol, Asynchronous Timeliness

1 Introduction

In the actual e-commerce place users can access Internet in order to make purchases, bid for products, buy cinema tickets, etc. As it is widely known, one of the main factors for achieving successful e-commerce procedures is *trust*. We can analyze the example of *eBay*, which is possibly the most popular bidding site in the Internet. A strong reason for this is the reputation system being used, which establishes certain trust among users.

However, it is obvious that this will not last forever. As more companies come into play and *peer to peer* (P2P) networks spread over the Internet, it is not easy to guarantee beforehand trust among participants. All the users will demand more and more control and security in their communications and transactions. Therefore, there is a need for the existence of fair electronic exchange since the basis of commercial transactions is the exchange of items.

Fair exchange protocols are mechanisms to ensure that items held by two parties are exchanged without one party obtaining an advantage. The most important applications of fair exchange protocols are *certified electronic mail* and *electronic contract signing*.

Fairness of an exchange would not be too difficult to guarantee if both parties involved in the transaction behave honestly or if, alternatively, an online *trusted third party* (TTP) is used during the exchange. The first situation does not hold because most of time parties do not know

each other, so there is no implicit trust. On the other hand, protocols for the second situation are inefficient because the TTP must be part of every execution, and also expensive because the TTP charges for every single run of the protocol. Additionally, from the legal perspective, the TTP will be somehow involved in the liability of every exchange between parties.

Our objective is to focus on fair exchange protocols that use a TTP only in those cases in which an exception occurs (i.e., a network communication failure or a party's misbehavior). In the literature, these protocols have been called *optimistic fair exchange* protocols (Asokan, Schunter and Waidner, 1997; Asokan, Shoup and Waidner, 2000; González-Deleito and Markowitch, 2001; Zhou and Gollmann, 1997; Zhou, 2001). In this paper, we use the new protocols for optimistic fair exchange proposed by Micali in (Micali, 2003) as a foundation, and extend those schemes with asynchronous timeliness and also for a multi-party scenario.

The rest of this paper is organized as follows. We elaborate on the fair exchange scenarios and their properties in Section 2. Then, we briefly explain Micali's solutions for certified email and contract signing in Section 3. After that, we extend Micali's protocols for asynchronous timeliness in Section 4 and for multi-party scenario in Section 5. Section 6 concludes the paper.

2 Electronic Exchange

Many commercial transactions can be modelled as a sequence of exchanges of electronic goods involving two or more parties. An exchange between two parties begins with an agreement regarding what item will be the contribution of each party to the exchange. Neither party knows the other's item, but will recognize it whenever the party gets it. Therefore, the security problem we are facing is how to exchange the items without giving advantage to any participating entity. This could be achieved through an online TTP, but as stated in the previous section, this is not convenient. On the other hand, as suggested in (Pagnia and Gärtner, 1999), it is impossible to reach a fair state after an exchange in all the possible cases without the help of a trusted third party.

We sketch the basic fair exchange scenario in Figure 1. In the initial state both parties, Alice and Bob, own some kind of information about the item expected (e.g., the format). Since that moment on, and until the last step in which Bob's item is sent to Alice, we face an unfair state in the scenario. Therefore, the protocol must be designed in such a way that the unfair situation is limited even in case of a communication channel failure or a transacting party's misbehavior. This is precisely the case in which the participation of a third party is unavoidable.

2.1 Security Properties

We identify several requirements in a typical two-party fair electronic exchange service. Some of them may be optional depending on the application the fair exchange service is running over. Although fairness is probably the most important one, in some scenarios there are other properties that are needed for the correct behavior of the application:

- *Fairness*: An electronic exchange protocol provides fairness if neither party can gain an advantage by quitting prematurely or otherwise misbehaving during the protocol. At the

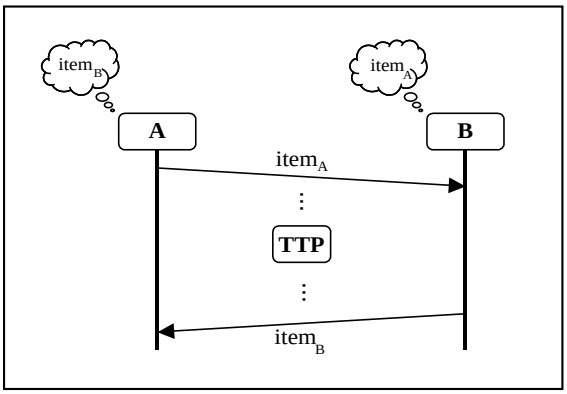

Figure 1: Basic Fair Exchange Scenario

end of the protocol (see Figure 1), either Alice gets $item_B$ and Bob receives $item_A$, or none of them gets any valuable item.

- *Timeliness*: An electronic exchange protocol provides timeliness if any of the participating entities has the ability to reach the end of the protocol in a finite amount of time without loss of fairness.
- *Confidentiality*: An electronic exchange protocol provides confidentiality if none but the intended parties can get access to the (plaintext) items sent during the protocol.
- *Non-repudiation*: An electronic exchange protocol provides non-repudiation if neither Alice nor Bob can deny having sent the item.

Considering that either Alice or Bob can be dishonest, the communication channel between them is not important. We assume that the communication channel between the TTP and the users is not permanently broken (i.e., the messages sent to/from the TTP will reach its destination in a finite amount of time).

3 Micali's Optimistic Protocols

Here we briefly describe Micali's optimistic protocols for *certified electronic mail* (CEM) and *electronic contract signing* (ECS) (Micali, 2003). As Micali explained, each user in the system has a unique identifier. We denote Alice's identifier by A, Bob's by B, the invisible trusted party by TTP (and the post office by PO in the case of CEM). We assume that Alice, Bob and the TTP can all sign messages using a digital signature scheme non-existentially forgeable by an adaptive chosen message attack. Party X's signature of a message M is denoted by $SIG_X(M)$, and we assume, for convenience, that M is always retrievable from $SIG_X(M)$. We also assume that Alice, Bob and the TTP can also encrypt messages by means of a public-key encryption algorithm that is secure against adaptive chosen ciphertext attack. By $E_X(M)$, we denote encryption of a message M with X's public key. [1]

[1]For simplicity, we assume that messages are encrypted directly with a public-key algorithm. But, according to standard practice, we could first encrypt a message M conveniently with some symmetric (session) key K, and then encrypt K with a public-key algorithm.

As we can see in the following protocols, even when the TTP intervenes only in exceptional cases, its participation is *transparent*, i.e., if the protocol can be resolved successfully for both parties, evidences obtained by them do not change.

3.1 A Protocol for Fair CEM with Cut-Off Time

Although Micali explained different fair CEM protocols in (Micali, 2003), we describe here the most complete one that supports confidentiality, fairness and timeliness. Because it is an optimistic protocol, if both parties behave honestly, the TTP (which is also referred as the post office PO here) will not be involved. Before sending the plaintext message M to Bob, Alice computes a secret Z protected with the TTP's encryption key as $Z = E_{PO}(A, B, E_B(M))$. To reach timeliness, Micali proposed a cut-off time solution, where Alice chooses a time t after which the TTP should not help Bob in the conclusion of the protocol.

1. $A \rightarrow B \;\; : \;\; SIG_A(t, Z)$
2. $B \rightarrow A \;\; : \;\; SIG_B(Z)$
3. $A \rightarrow B \;\; : \;\; E_B(M)$

Whenever Bob reaches step 1 and verifies Alice's signature, he must extract the cut-off time t and estimate whether he will have enough time to contact the PO in case of Alice's misbehavior or channel failure. t_D denotes the maximum possible time discrepancy that Bob believes may exist between his clock and that of the PO. If Bob receives step 1 in time t_B (i.e., Bob's local time) such that $t_B + t_D$ is greater than or equal to t, then Bob halts; otherwise he proceeds to step 2. After verifying Bob's signature, Alice sends the message M to Bob at step 3.

After replying at step 2, if Bob does not get the message within a reasonable amount of time, or $Z = E_{PO}(A, B, E_B(M))$ does not hold, Bob contacts the PO in a *resolve* sub-protocol:

$1'. \; B \rightarrow PO \;\; : \;\; SIG_A(t, Z), SIG_B(Z)$
$2'.$ IF ($t_{PO} < t$) AND (valid signatures)
$PO \rightarrow B \;\; : \;\; X$
$PO \rightarrow A \;\; : \;\; SIG_B(Z)$

In this sub-protocol, the PO verifies whether Bob's request arrives before Alice's cut-off time and also whether both signatures are correct. If so, the PO decrypts Z with its private key and, if the result is a triplet consisting of *A*, *B*, and an unknown string X, it sends X to Bob and forwards Bob's signature to Alice.

3.2 A Protocol for Fair ECS with Cut-Off Time

Although Micali explained different fair ECS protocols in (Micali, 2003), here we only describe the most complete one that supports confidentiality, fairness and timeliness. Assume that Alice and Bob have already negotiated a would-be contract C and now wish to execute it fairly. Then, Alice chooses a random message M, and uses the TTP's public encryption key to compute a value $Z = E_{TTP}(A, B, M)$ as if she wanted to send M to Bob. Alice is committed to C if Bob has both (i) her own signature of (C, Z) and (ii) M. Bob is committed to C if Alice has both (i) his own signature of (C, Z) and (ii) his own signature of Z. The description of this protocol is similar to the previous one:

1. $A \rightarrow B$: $SIG_A(t, C, Z)$
2. $B \rightarrow A$: $SIG_B(C, Z), SIG_B(Z)$
3. $A \rightarrow B$: M

After replying at step 2, if Bob does not get the message M within a reasonable amount of time, or $Z = E_{TTP}(A, B, M)$ does not hold, Bob contacts the TTP in a *resolve* sub-protocol:

$1'$. $B \rightarrow TTP$: $SIG_A(t, C, Z), SIG_B(C, Z), SIG_B(Z)$
$2'$. IF ($t_{TTP} < t$) AND (valid signatures)
$TTP \rightarrow B$: M
$TTP \rightarrow A$: $SIG_B(C, Z), SIG_B(Z)$

In this sub-protocol, the TTP verifies whether Bob's request arrives before Alice's cut-off time and also whether all signatures are correct. If so, the TTP decrypts Z with its private key and, if the result is a triplet consisting of *A*, *B*, and a string M, it sends M to Bob and forwards Bob's signatures to Alice.

There exists a flaw in Micali's original ECS protocol, which could lead to the breach of fairness (Feng Bao and Zhu, 2004).

4 Extension to Fair CEM with Asynchronous Timeliness

We believe that a cut-off time is not the best solution for a timeliness property. Thus, in this section we propose a different solution, *asynchronous timeliness* (i.e., either party can finish the protocol at any time without loss of fairness). In Micali's proposal, even if Bob approximately calculates in each run the time to contact the PO, there can be always a situation in which the PO is unaccessible for longer. In such a case Bob will not get the item from the fair exchange protocol and it will be difficult to figure out who bears the responsibility for the breach of fairness.

We introduce a new *cancel* sub-protocol. With the *resolve* sub-protocol, Bob will be able to finish the protocol any time. In this way, if Alice does not want to wait for the resolution of the protocol she can abort it with *cancel* sub-protocol any time too. In the description below, $X \leftarrow PO : item$ means that entity X accesses an item on PO's server, allowing PO getting rid of the responsibility of communication with X.

The revised *main* protocol (and the only one needed in case both parties behave and no error occurs in the communication channel) is as follows:

1. $A \rightarrow B$: Z
2. $B \rightarrow A$: $SIG_B(Z)$
3. $A \rightarrow B$: $E_B(M)$

The revised *resolve* sub-protocol is as follows, which will be requested by Bob under the same conditions as the original one:

$1'$. $B \rightarrow PO$: $h(Z), SIG_B(Z)$
$2'$. $B \leftarrow PO$: IF cancelled THEN $SIG_A(cancel, Z)$
ELSE $E_B(M)$

The new *cancel* sub-protocol is as follows:

$1'.\ A \rightarrow PO$: $h(Z), SIG_A(cancel, Z)$
$2'.\ A \leftarrow PO$: IF resolved THEN $SIG_B(Z)$
ELSE *ack*

We use a one-way hash function $h(Z)$ to facilitate the search of the item Z by the PO when one of the participating entities requests the service, thus acting as a label. Because there is no cut-off time for Bob, Alice can abort the protocol as desired. As we can see, the PO provides access to the items that either of the parties needs. The parties can access the data at any time and it is not the responsibility of the PO whether the users get the messages they expect. The PO is not stateless and needs to maintain (within a reasonable amount of time) a table with entries $(h(Z), state)$ and stores the signatures for serving the users. Therefore, the server implementing the PO must secure this information (although confidentiality is not needed).

Although in (Micali, 2003) there is no explicit definition of the dispute resolution process, we think it is in general necessary for any fair exchange protocol. In this process both parties must agree that an arbitrator will evaluate the final outcome of the protocol based on the evidence provided by the users. Consequently, if Bob denies having received a message in a CEM protocol run, Alice should provide $(Z, SIG_B(Z))$ and the arbitrator settles that Alice sent the message M to Bob if

i) $Z = E_{PO}(A, B, E_B(M))$ holds;
ii) Bob's signature on Z is valid;
iii) Bob cannot provide $SIG_A(cancel, Z)$.

We assume a deterministic public encryption algorithm, or otherwise, Alice cannot discard the random seed if a non-deterministic public encryption algorithm (e.g., the ElGamal cryptosystem (ElGamal, 1985)) is used.

Our approach can be easily applied to extend fair ECS protocol with asynchronous timeliness.

5 Extension to Multi-Party Fair CEM

There are scenarios in which the participation of multiple entities can result in an important improvement. CEM and ECS are two of them. We can easily figure out applications in which sending e-mail to several users, or having the contract agreed by various participants, are feasible and useful. This is especially true in CEM. In this section we propose a new multi-party CEM protocol based on Micali's two-parties CEM protocol. We provide the possibility for the sender to distribute in a certified manner a message M to several recipients. Some additional notation in the protocol description is

- B : a set of intended recipients.
- $Header$: a header indicating in which position the recipient has to look for its information (e.g. $[B_i, i]$).
- M : message being sent from Alice to the recipients B.
- $A \Rightarrow B : X$: Alice broadcasts string X to group B.
- B' : a subset of B that replied to Alice with the evidence of receipt.
- $B'' = B - B'$: a subset of B (in plaintext) with which Alice wants to cancel the exchange.

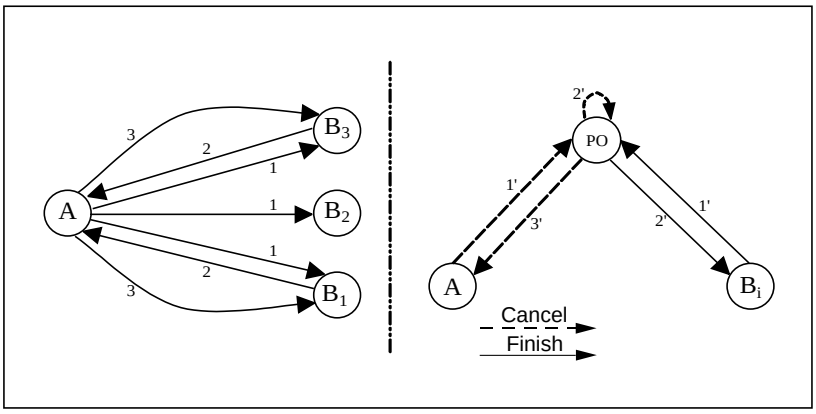

Figure 2: Multi-Party Fair CEM Protocol

- $B''\_finished$: a subset of B'' that have finished the exchange with the ***finish*** sub-protocol.
- $B''\_cancelled = B'' - B''\_finished$: a subset of B'' with which the exchange has been cancelled by the PO.
- $u_B = u_{B_1}, u_{B_2}, ...$: concatenation of public keys from group B.
- $E_B(M) = E_{B_1}(M), E_{B_2}(M), ...$: an encryption concatenation of M for group B.
- $Z = E_{PO}(A, B, E_B(M))$: a secret Z protected with the PO's encryption key.

The PO will participate, if requested by any entity, in a mutually exclusive way (i.e., atomic execution of the sub-protocols for each user). Here, we describe the protocol (see Figure 2). The *main* protocol executed by the final entities is:

$$
\begin{array}{lcl}
1.\ A \Rightarrow B & : & Header, u_B, Z \\
2.\ B_i \rightarrow A & : & SIG_{B_i}(u_{B_i}, Z) \text{ where each } B_i \in B \\
3.\ A \Rightarrow B' & : & E_{B'}(M)
\end{array}
$$

In step 1, Alice sends the secret Z including all the recipients' public keys such that if any recipient does not agree with its public encryption key (e.g., the corresponding public key certificate has been revoked), then it stops the protocol. Otherwise, after verifying the data obtained, the recipient sends evidence of receipt to Alice at step 2, and she sends the (encrypted) message M at step 3 to the set of recipients who replied.

If Alice did not receive a correct message 2 from some of the recipients B'', she may initiate the following *cancel* sub-protocol:

$$
\begin{array}{lcl}
1'.\ A \rightarrow PO & : & h(Z), SIG_A(cancel, B'', Z) \\
2'.\ PO & & \text{FOR (all } B_i \in B'') \\
 & & \text{IF } (B_i \in B''\_finished) \text{ THEN retrieves } SIG_{B_i}(u_{B_i}, Z) \\
 & & \text{ELSE appends } B_i \text{ into } B''\_cancelled \\
3'.\ A \leftarrow PO & : & \text{all retrieved } SIG_{B_i}(u_{B_i}, Z),\ SIG_{PO}(B''\_cancelled, Z)
\end{array}
$$

In this case Alice communicates the PO its intention of revoking the protocol with entities contained in B''. After verifying Alice's cancel request, the PO checks which entities previously

resolved the protocol and gets their proofs of receipt. Then, the PO generates an evidence of cancellation for the rest of entities and includes everything in a message destined to Alice.

If some recipient B_i did not receive the message 3 or it was not valid, B_i may initiate the following ***finish*** sub-protocol:

$$
\begin{array}{lcl}
1'.\ B_i \rightarrow PO & : & Header, h(Z), u_{B_i}, SIG_{B_i}(u_{B_i}, Z) \\
2'.\ B_i \leftarrow PO & : & \text{IF } (B_i \in B''\_cancelled) \text{ THEN } SIG_{PO}(B''\_cancelled, Z) \\
 & & \text{ELSE } \{E_{B_i}(M) \\
 & & \text{appends } B_i \text{ into } B''\_finished \text{ and stores } SIG_{B_i}(u_{B_i}, Z)\}
\end{array}
$$

The recipient sends to the PO all the information that it has already got from Alice along with its evidence of receipt. If this entity does not belong to the group of entities with which Alice cancelled the exchange, the PO verifies all the information (digital signatures) and decrypts Z. It also stores $SIG_{B_i}(u_{B_i}, Z)$. Note that if the protocol has been cancelled, it should be impossible for the recipient to cheat the PO in a way that the PO reveals the message for that protocol run. For that reason, the PO must verify the recipient's signature as well as integrity of Z in the first step.

Otherwise, the PO sends a cancellation evidence to the recipient such that the latter can easily demonstrate to an arbitrator that the exchange was cancelled in case a dispute arises.

If B_i denies having received M, Alice can present evidence $Z, E_B(M), SIG_{B_i}(u_{B_i}, Z)$, $SIG_{PO}(B''\_cancelled, Z)$ and the arbitrator settles that the recipient received the message M from Alice if

i) $E_{B_i}(M)$ computed with u_{B_i} belongs to $E_B(M)$;
ii) $Z = E_{PO}(A, B, E_B(M))$ holds;
iii) B_i's signature on Z and its encryption public key is valid;
iv) PO's signature on $SIG_{PO}(B''\_cancelled, Z)$ and $B_i \notin B''\_cancelled$.

Alice will succeed on the dispute if all the above checks are positive. If all the checks but the last are positive and she cannot present evidence of cancellation, then the arbitrator must further interrogate B_i. If the latter cannot present $SIG_{PO}(B''\_cancelled, Z)$ in which $B_i \in B''\_cancelled$, Alice also wins the dispute. Otherwise, B_i can repudiate having received the message M. Therefore, evidence provided by the PO is ***self-contained***, that is, the PO need not be contacted in case a dispute arises regarding the occurrence of a cancellation sub-protocol launched by Alice.

6 Conclusions

In this paper we briefly described Micali's solution to certified electronic mail and electronic contract signing, pointing out how the timeliness property proposed in the original paper may not be enough for totally securing the termination of the protocol for both parties. Although a new *cancel* sub-protocol is added, the *main* protocol (the only one executed in case no abnormal situations appear) remains unchanged. The additional messages included augment the

timeliness property in an efficient and elegant way.

At the same time a multi-party extension for the distribution of the same mail (message) to several entities in a certified manner is made in such a way that only those who reply with evidence of receipt will obtain the message. Timeliness is also considered here and though the PO has to deal with several entities in case any of them launches a sub-protocol, no significant complexity is introduced.

In both approaches, the dispute resolution process is defined because in case a dispute arises among the parties, the process must be clear enough for an arbitrator to resolve the exchange according to the evidence provided by the participating entities.

References

Asokan, N., Matthias Schunter and Michael Waidner. 1997. Optimistic protocols for fair exchange. In *Proceedings of the 4th ACM conference on Computer and Communications security*. ACM Press pp. 7–17.

Asokan, N., Victor Shoup and Michael Waidner. 2000. "Optimistic Fair Exchange of Digital Signatures." *IEEE Journal on Selected Areas in Communications* 18(4):593–610.

ElGamal, T. 1985. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *IEEE Transactions on Information Theory*. Vol. 31 pp. 469–472.

Feng Bao, Guilin Wang, Jianying Zhou and Huafei Zhu. 2004. Analysis and improvement of Micali's fair contract signing protocol. In *Proceedings of the 9th Australasian Conference on Information Security and Privacy*. Lecture Notes in Computer Science Springer-Verlag.

González-Deleito, N. and O. Markowitch. 2001. An Optimistic Multi-party Fair Exchange Protocol with Reduced Trust Requirements. In *Proceedings of the 4th International Conference on Information Security and Cryptology*. Vol. 2288 of *Lecture Notes in Computer Science* Springer-Verlag pp. 258–267.

Micali, Silvio. 2003. Simple and fast optimistic protocols for fair electronic exchange. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*. ACM Press pp. 12–19.

Pagnia, Henning and Felix C. Gärtner. 1999. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02 Darmstadt University of Technology, Department of Computer Science Darmstadt, Germany: .

Zhou, Jianying. 2001. Achieving fair non-repudiation in electronic transactions. In *Journal of Organizational Computing and Electronic Commerce*. Vol. 11 Lawrence Erlbaum Associates pp. 253–267.

Zhou, Jianying and D. Gollmann. 1997. An Efficient Non-repudiation Protocol. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press pp. 126–132.