Optimized Multi-Party Certified Email Protocols *

Jianying Zhou\textsuperscript{1}, Jose Onieva\textsuperscript{2}, and Javier Lopez\textsuperscript{2}

\textsuperscript{1} Institute for Infocomm Research  
21 Heng Mui Keng Terrace  
Singapore 119613  
jyzhou@i2r.a-star.edu.sg

\textsuperscript{2} Computer Science Department  
E.T.S. Ingenieria Informatica  
University of Malaga  
Spain, 29071-Malaga  
\{onieva,jlm\}@lcc.uma.es

Abstract. As a value-added service to deliver important data over the Internet with guaranteed receipt for each successful delivery, certified email has been discussed for years and a number of research papers appeared in the literature. But most of them deal with the two-party scenarios, i.e., there are only one sender and one recipient. In some applications, however, the same certified message may need to be sent to a set of recipients. In this paper, we present two optimized multi-party certified email protocols. Both of them have three major features. (1) A sender could notify multiple recipients of the same information while only those recipients who acknowledged are able to get the information. (2) Both the sender and the recipients can end a protocol run at any time without breach of fairness. (3) The exchange protocols are optimized, each of which has only three steps, and the TTP will not be involved unless an exception (e.g., a network failure or a party's misbehavior) occurs.

Keywords: certified email, non-repudiation, multi-party protocol

1 Introduction

Email has grown from a tool used by a few academics on the Arpanet to a ubiquitous communications tool. Certified email is a value-added service of ordinary email, in which the sender wants to obtain a receipt from the recipient. In addition, fairness is usually a desirable requirement thus the recipient gets the mail content if and only if the sender obtains a receipt.

Certified email has been discussed for years, and a number of research papers appeared in the literature \cite{1,2,4,11,14–16}. But most of them deal with the

* Major results have been published at ICICS’04 \cite{13} and INC’04 \cite{12}. 
two-party scenarios, i.e., there are only one sender and one recipient. In some applications, however, the same certified message may need to be sent to a set of recipients. Multi-party certified email protocols were first proposed by Markowitch and Kremer, using an on-line trusted third party [7], or an off-line trusted third party [8]. In ISC’02, Ferrer-Gomila et. al presented a more efficient multi-party certified email protocol [6]. However, this protocol suffers from a number of serious security problems [13].

In this paper, we present two optimized multi-party certified email protocols, based on Ferrer-Gomila et. al’s protocol [6] and Micali’s protocol [10], respectively. They have three major features: (1) A sender could notify multiple recipients of the same information while only those recipients who acknowledged are able to get the information; (2) Both the sender and the recipients can end a protocol run at any time without breach of fairness; (3) The exchange protocols are optimized, each of which has only three steps, and the trusted third party will not be involved unless an exception (e.g., a network failure or a party’s misbehavior) occurs.

The rest of the paper is organized as follows. We first review Ferrer-Gomila et. al’s multi-party certified email protocol in Section 2, then present a modified version that overcomes its security flaws and weaknesses in Section 3. After that, we extend Micali’s two-party certified email protocol to a multi-party scenario with asynchronous timeliness in Section 4. Finally, we conclude the paper in Section 5.

2 Analysis of FPH Protocol

A multi-party certified email protocol was presented in [6]. (We call it FPH protocol in this paper.) The sender $A$ of a certified email and a set of recipients $B$ exchange messages and non-repudiation evidence directly, with the exchange sub-protocol. If the exchange sub-protocol is not completed successfully, a trusted third party $TTP$ will be invoked, either by $A$ with the cancel sub-protocol, or by $B$ with the finish sub-protocol.

FPH protocol is efficient, but suffers from a number of serious security problems.

2.1 FPH Protocol

Here, we give a brief description of FPH protocol with the same notation used in the original paper.

- $X, Y$: concatenation of two messages $X$ and $Y$.
- $H(X)$: a collision-resistant one-way hash function of message $X$.
- $E_K(X)$ and $D_K(X)$: symmetric encryption and decryption of message $X$.
- $P_U(X)$ and $P_U^{-1}(X)$: asymmetric encryption and decryption of message $X$.
- $S_U(X)$: principal $U$’s digital signature on message $X$.
- $U \rightarrow V$: entity $U$ sends message $X$ to entity $V$. 
- $A \Rightarrow B$: $X$: entity $A$ sends message $X$ to a set of entities $B$.
- $M$: certified message to be sent from $A$ to the set $B$.
- $K$: symmetric key selected by $A$.
- $c = E_K(M)$: ciphertext of message $M$, encrypted with key $K$.
- $k_T = P_T(K)$: key $K$ encrypted with the TTP’s public key.
- $h_A = S_A(H(c), B, k_T)$: first part of evidence of origin for every recipient $B_i \in B$.
- $h_B = S_B(H(c), k_T)$: evidence of receipt for $A$.
- $k_A = S_A(K, B')$: second part of evidence of origin for $B_i \in B'$.
- $k_A' = S_T(K, B_i)$: alternative second part of evidence of origin for $B_i$.
- $h_{AT} = S_A(H(c), k_T, h_A, B'')$: evidence that $A$ has demanded the TTP’s intervention to cancel the exchange sub-protocol with $B_i \in B''$.
- $h_{B,T} = S_B(H(c), k_T, h_A, h_B, h_{AT})$: evidence that $B_i$ has demanded the TTP’s intervention to finish the exchange sub-protocol with $A$.

The exchange sub-protocol is as follows, where $B_i \in B$ and $B'$ is a subset of $B$ that have replied message 2.

1. $A \Rightarrow B : c, k_T, B, h_A$
2. $B_i \Rightarrow A : h_{B_i}$
3. $A \Rightarrow B' : K, B', h_A$

If $A$ did not receive message 2 from some of the recipients $B''$, $A$ may initiate the following cancel sub-protocol, where $B'' = B - B'$.

1'. $A \rightarrow TTP : H(c), k_T, B, h_A, B'', h_{AT}$
2'. $TTP$:
   - FOR (all $B_i \in B''$)
     - IF ($B_i \in B''.finished$) THEN retrieves $h_{B_i}$
     - ELSE appends $B_i$ into $B''.finished$
3'. $TTP \rightarrow A$:
   - all retrieved $h_{B_i}$, $B''.finished$,
   - $S_T(\text{"cancelled"}, B''.\text{cancelled}, h_A)$, $S_T(B''.\text{finished})$

If some recipient $B_i$ did not receive message 3, $B_i$ may initiate the following finish sub-protocol.

2'. $B_i \rightarrow TTP : H(c), k_T, B, h_A, h_{B_i}, h_{B,T}$

IF ($B_i \in B''.\text{cancelled}$) 3'. $TTP \rightarrow B_i : S_T(\text{"cancelled"}, h_{B_i})$
ELSE 4'. $TTP$:
   - appends $B_i$ into $B''.\text{finished}$, and stores $h_{B_i}$

Dispute of Origin
In the case of dispute of origin, a recipient $B_i$ claims that he received $M$ from $A$ while $A$ denies having sent $M$ to $B_i$. $B_i$ has to provide $M, c, K, k_T, B, h_A$ and $B', h_A$ (or $k_T'$) to an arbiter. The arbiter will check

(O-1) if $h_A$ is $A$’s signature on $(H(c), B, k_T)$, and $B_i \in B$;
(O-2) if $k_A$ is $A$’s signature on $(K, B')$ and $B_i \in B'$, or if $k'_T$ is the TTP’s signature on $(K, B_i)$;
(O-3) if the decryption of $c$ (i.e., $D_K(c)$) is equal to $M$.

$B_i$ will win the dispute if all of the above checks are positive.

Dispute of Receipt

In the case of dispute of receipt, $A$ claims that a recipient $B_i$ received $M$ while $B_i$ denies having received $M$. $A$ has to provide $M, c, K, k_T, h_{B_i}$ to an arbiter. The arbiter will check

(R-1) if $h_{B_i}$ is $B_i$’s signature on $(H(c), k_T)$;
(R-2) if $k_T$ is the encryption of $K$ with the TTP’s public key;
(R-3) if the decryption of $c$ (i.e., $D_K(c)$) is equal to $M$.

If one of the above checks fails, $A$ will lose the dispute. Otherwise, the arbiter must further interrogate $B_i$. If $B_i$ is able to present a cancellation token $S_T(\text{"cancelled"}, h_{B_i})$, it means that $B_i$ had contacted the TTP and was notified that $A$ had executed the cancel sub-protocol. Then $A$ will lose the dispute as well. If all of the above checks are positive and $B_i$ cannot present the cancellation token, $A$ will win the dispute.

2.2 Vulnerabilities

V-1. Who is TTP

In FPH protocol, it is not expressed explicitly that all users share a unique TTP. There may be a number of TTPs and the sender may have the freedom to select the TTP, which may not be the one that the recipient is aware of.

In the exchange sub-protocol, the sender $A$ needs to select a TTP and uses the TTP’s public key to generate $k_T$. However, $A$ did not provide the identity of the TTP in message 1. If $A$ terminates the protocol without sending message 3, it is very likely that the recipient $B_i$ is unable to identify which TTP should be invoked to launch the finish sub-protocol. That means $B_i$ can neither obtain $M$ by decrypting $c$ with $K$ from the TTP nor get $S_T(\text{"cancelled"}, h_{B_i})$ to prove cancellation of receiving $M$.

On the other hand, $A$ can use $h_{B_i}$ to prove that $B_i$ has received $M$ when $B_i$ cannot present the cancellation token $S_T(\text{"cancelled"}, h_{B_i})$.

There are two possible solutions to this problem. We might assume that all users share a single TTP. Then $B_i$ can always initiate the finish sub-protocol with this TTP. Obviously, this assumption is unrealistic in the actual deployment.

Alternatively, $A$ should specify the TTP explicitly in message 1. Then, $B_i$ could decide whether or not to accept $A$’s choice of this TTP. If not, $B_i$ can simply terminate the exchange sub-protocol. Otherwise, $B_i$ should include the identity of the TTP in $h_{B_i}$ when replying message 2. A modified exchange sub-protocol is as follows, where the modified parts are underscored.
\[ h_A = S_A(H(c), B, TTP, k_T) \]
\[ h_{B_i} = S_{B_i}(H(c), TTP, k_T) \]

1. \( A \Rightarrow B : c, k_T, B, TTP, h_A \)
2. \( B_i \rightarrow A : h_{B_i} \)
3. \( A \Rightarrow B' : K, B', k_A \)

If \( A \) cheats at Step 1 in the revised exchange sub-protocol by encrypting \( K \) with a public key of the TTP but indicating to \( B_i \) as the TTP, \( A \) will not be able to get the valid evidence of receipt. When \( A \) presents \( M, c, k_{T1} = P_{T1}(K), h_{B_i} = S_{B_i}(H(c), TTP, k_{T1}) \), \( K \) to an arbiter, the arbiter will identify the TTP in \( h_{B_i} \) and use the TTP’s public key to verify whether encryption of \( K \) equals \( k_{T1} \), which obviously leads to the failure of requirement (R-2). That means \( A \) cannot win in the dispute of receipt.

Therefore, the above modified exchange sub-protocol can prevent the sender’s attack on the use of a TTP that the recipient is unaware of.

V-2. How can B verify evidence of origin along

In FPH protocol, it is claimed that an arbitrary asymmetric cryptography could be used as a building block. Unfortunately, this may not be true.

In the exchange sub-protocol, the sender \( A \) may send a different key \( K_1 \) and \( k_{A1} = S_A(K_1, B') \) instead of \( K \) and \( k_A \) at Step 3. Then, the recipient \( B_i \) believes that the exchange is successful and \( B_i \) holds the evidence \( h_A \) and \( k_{A1} \) which can prove \( M1 = D_{K1}(c) \) is from \( A \). On the other hand, \( A \) can use \( h_{B_i} \) to prove that \( B_i \) received \( M \).

To protect against this attack, \( B_i \) needs to check whether \( K \) received at Step 3 is consistent with \( k_T \) received at Step 1. If not, \( B_i \) needs to initiate the finish sub-protocol.

Suppose a non-deterministic public encryption algorithm (e.g., the ElGamal cryptosystem [5]) is used, and \( A \) has discarded the random seed used during the encryption phase. Then, even if \( B_i \) holds \( k_T, K \), and the TTP’s public key, \( B_i \) cannot verify whether \( k_T \) is the encryption of \( K \) with the TTP’s public key.

Of course, \( B_i \) may always initiate the finish sub-protocol to either get \( K \) (and thus \( M \)) or get \( S_T(\text{“cancelled”}, h_{B_i}) \) from the TTP. However, the merit of FPH protocol is that the TTP is invoked only in the abnormal situation (i.e., either \( A \) did not receive message 2 or \( B \) did not receive message 3). If the TTP is involved in every protocol run, it becomes an on-line TTP, and the protocol will be designed in a totally different way.

A straightforward solution is to ask \( A \) to supply the random seed with \( K \) in message 3 thus \( B \) can verify \( K \) in \( P_T(K) \) directly.

Alternatively, the problem could be solved if \( A \) provides \( H(K) \) in message 1, and \( B_i \) includes \( H(K) \) in \( h_{B_i} \) so that \( B_i \) is only liable for receipt of a message

\footnote{If the algorithm is non-deterministic, \( A \) needs to provide the random seed used in encryption so that the arbiter can verify whether \( k_{T1} \) is the encryption of \( K \) with the TTP’s public key. Otherwise, the TTP has to be invoked to decrypt \( k_{T1} \) first.}
decrypted with the key that is consistent in $H(K)$ and $k_T$. The *exchange* sub-
protocol is further modified as follows.

$$\begin{align*}
    h_A &= S_A(H(c), B, TTP, H(K), k_T) \\
    h_{B_i} &= S_{B_i}(H(c), TTP, \overline{H(K)}, k_T)
\end{align*}$$

1. $A \Rightarrow B : c, H(K), k_T, B, TTP, h_A$
2. $B_i \rightarrow A : h_{B_i}$
3. $A \Rightarrow B' : K, B', k_A$

Two additional checks should be taken in the settlement of disputes.

(O-4) $K$ certified in $k_A$ or $k_T'$ must match $H(K)$ certified in $h_A$.
(R-4) $H(K)$ and $k_T$ certified in $h_{B_i}$ must match, i.e., $H(P_T^{-}(k_T)) = H(K)$.

If $A$ cheats at Step 1 in the revised *exchange* sub-protocol by providing $k_{T1} = P_T(K1)$ and $h_A = S_A(H(c), B, TTP, H(K), k_{T1})$, $B_i$ will reply with $h_{B_i} = S_{B_i}(H(c), TTP, \overline{H(K)}, k_{T1})$. Then, no matter $A$ sends $K$ or $K1$ at Step 3, $A$ cannot use $h_{B_i}$ to prove either $B_i$ received $M = D_K(c)$ or $B_i$ received $M1 = D_{K1}(c)$. The verification on $h_{B_i}$ will fail when $H(P_T^{-}(k_{T1})) \neq H(K)$.

If $A$ cheats only at Step 3 by providing $K1$ and $k_{A1} = S_A(K1, B')$, $B_i$ can detect the cheat by checking whether $H(K1) = H(K)$ where $H(K)$ is received at Step 1. If the check fails, $B$ should initiate the *finish* sub-protocol. Then, there are two possibilities. If $A$ did not cancel the exchange, $B_i$ will receive $K$ and thus $M = D_K(c)$. If $A$ has cancelled the exchange, $B_i$ will receive $S_T(\text{"cancelled", } h_{B_i})$. In either case, $A$ cannot get any advantage when $A$ wants to use $h_{B_i}$ to settle the dispute.

With the above modification of the protocol, the restriction on the use of an asymmetric algorithm for public encryption could be removed. Moreover, this modification could also stop another attack described below.

**V-3. How to stop B misusing evidence of origin**

In FPH protocol, it is assumed that the elements to link messages of an exchange is omitted in order to simplify the explanation. However, as these elements are critical to the protocol security and not so obvious to handle, they cannot be omitted in any way.

With the original definition of $h_A$ and $k_A$ (or $k_T'$), the recipient $B_i$ can misuse the evidence in settling disputes of origin. Suppose $B_i$ received $h_{A1} = S_A(H(c1), B, k_{T1})$, $k_{A1} = S_A(K1, B')$, and the related messages in the first protocol run. $B_i$ also received $h_{A2} = S_A(H(c2), B, k_{T2})$, $k_{A2} = S_A(K2, B')$, and the related messages in the second protocol run. If the protocol is designed correctly, $B_i$ can only use $h_{A1}$ and $k_{A1}$ to prove that $M1 = D_{K1}(c1)$ is from $A$, and use $h_{A2}$ and $k_{A2}$ to prove that $M2 = D_{K2}(c2)$ is from $A$.

Note that the original rules in settling disputes of origin do not check whether decryption of $k_T$ certified in $h_A$ equals $K$ certified in $k_A$ (or $k_T'$). Then, $B_i$ can use $h_{A1}$ and $k_{A2}$ to prove that $M3 = D_{K2}(c1)$ is from $A$, and use $h_{A2}$ and $k_{A1}$
to prove that $M4 = D_{K1}(c2)$ is from $A$. But the fact is that $A$ never sent $M3$ and $M4$.

With the modification given in V-2, such an attack could also be stopped. The evidence received by $B_i$ will be as follows.

- $h_{A1} = S_A(H(c1), B, TTP, H(K1), k_{T1})$ and $k_{A1} = S_A(K1, B')$ in the first protocol run, and
- $h_{A2} = S_A(H(c2), B, TTP, H(K2), k_{T2})$ and $k_{A2} = S_A(K2, B')$ in the second protocol run.

If $B_i$ presents $h_{A1}$ and $k_{A2}$ to claim that $M3 = D_{K2}(c1)$ is from $A$, the arbiter will find that the hash of $K2$ certified in $k_{A2}$ does not equal $H(K1)$ certified in $h_{A1}$, and thus reject $B_i$’s claim. Similarly, $B_i$ cannot present $h_{A2}$ and $k_{A1}$ to claim that $M4 = D_{K1}(c2)$ is from $A$.

**V-4. How can TTP detect B’s cheating**

In FPH protocol, an intended recipient could collude with any party to cheat the TTP to decrypt $k_T$ and generate $k'_T$. This will lead to the breach of fairness.

Suppose $A$ initiates the exchange sub-protocol with a set of recipients $B$. Once $B_i \in B$ receives message 1 $(c, k_T, B, h_A)$ from $A$, it quits. Then $B_i$ asks a colluder $Z$ to generate $h_Z = S_Z(H(c'), B, k_T)$, and launches the finish sub-protocol by sending $H(c'), k_T, B, h_Z, h'_B, h'_B, h'_B, T$ to the TTP, where $h'_B = S_B(H(c'), k_T)$ and $h'_B = S_B(H(c'), k_T, h_Z, h'_B)$. The TTP only knows that $Z$ is exchanging with $B_i$ by examining $h_Z$. As $Z$ has not cancelled the exchange, the TTP sends $K, k'_T$ back to $B_i$. Then $B_i$ gets $M$, and holds evidence of origin $(h_A, k'_T)$ to prove that $M$ is from $A$!

On the other hand, if $A$ did not receive $h_B = S_B(H(c), k_T)$ from $B_i$ in the exchange sub-protocol, it launches the cancel sub-protocol. As the TTP did not find $B_i \in B''_{\text{finished}}$ related to the exchange with $A$, it only issues a cancel token to $A$.

To avoid the attack, the TTP needs to know who are the originator and the intended recipients of $k_T$. This can be achieved by making the further changes to $h_A$ and $h_B$ as follows.

$$
\begin{align*}
    h_A &= S_A(H(c), B, TTP, H(A, B, K), k_T) \\
    h_B &= S_B(H(c), TTP, H(A, B, K), k_T)
\end{align*}
$$

When the TTP receives a finish request, it can use $H(A, B, K)$ to identify the originator and recipients of $k_T$ and only releases $K$ to the intended recipients.

**V-5. How to prevent collusion among recipients**

In FPH protocol, fairness is a major security requirement. However, it is unfair to the sender $A$ if an intended recipient, after receiving message 1, intercepts message 3 without replying message 2. Although that recipient did not obtain valid evidence of origin in such a case, he got the message anyway without releasing evidence of receipt. This problem could be solved if the session key
$K$ in message 3 is encrypted in transmission. However, it does not work if two recipients collude.

Suppose $B_1$ and $B_2$ are two intended recipients specified by the sender $A$ (i.e., $B_1, B_2 \in B$). In the exchange sub-protocol, after receiving message 1, $B_1$ knows that $B_2$ is also a recipient, and vice versa. If they collude, $B_1$ can continue the protocol while $B_2$ terminates the protocol. At the end, $B_1$ receives the message $M$ and forwards it to $B_2$, but $A$ only holds the evidence that $B_1$ received the message $M$.

To prevent such an attack, we could re-define the set of intended recipients $B$ as follows.

$$B = P_{B_1}(B_1), P_{B_2}(B_2), \cdots$$

As each intended recipient’s identity is encrypted with their public key, when a recipient receives message 1, he can verify whether himself is an intended recipient included in $B$, but he does not know who are the other recipients. Then he is unable to identify a colluder \(^2\). The above change will not affect settling the dispute of origin on requirement (O-1).

Note that $B'$ also needs to be re-defined in the above way, but for a slightly different purpose. As $B'$ is a subset of $B$ that have replied message 2, all of them will receive the message $M$ and there is no need to prevent collusion among themselves. However, if $B'$ is transferred in clear text, an intended recipient $B_i$ that did not reply message 2 (i.e., $B_i \in B - B'$) could intercept message 3 and identify a colluder.

Further note that once a valid recipient receives the message $M$, it can always forward $M$ to any other parties. The above mechanism does not intend to stop such an active propagation. Instead, it only tries to make all the intended recipients anonymous to each other among themselves, thus it is hard for an intended recipient to seek a colluder (another intended recipient) to obtain the message without providing evidence of receipt to the sender.

### 2.3 Improvements

#### I-1. TTP need not keep evidence

In FPH protocol, in order to satisfy the requirement that the TTP is verifiable, the TTP must store evidence $h_{AT}$ of all protocol runs that the sender $A$ initiated the cancel sub-protocol. It will be used in the settlement of disputes which may arise sometime well after the end of a protocol run. If $A$ denies having cancelled an exchange when the recipient $B_i$ shows $S_T(\text{“cancelled”}, h_{B_i})$, the TTP should present $h_{AT}$ to prove that it did not misbehave. Obviously, this is a significant burden to the TTP.

A simple solution is to pass $h_{AT}$ to $B_i$ and include $h_{AT}$ in the cancellation token which becomes $S_T(\text{“cancelled”}, h_{B_i}, h_{AT})$. If a dispute arises, $B_i$ can (and

\(^2\) We assume that an intended recipient will not try to find a colluder by broadcasting message 1. This will expose the collusion to everyone.
should) use it to prove that the TTP cancelled the exchange demanded by A. Therefore, the TTP is not required to be involved in such a dispute and need not store the evidence for a long time.

I-2. B may not be involved in dispute of receipt

In FPH protocol, if there is a dispute of receipt, the recipient \( B \) has always to be interrogated on whether holding a cancellation token. This process could be optimized, thus \( B \) need not be involved unless the sender \( A \) did not invoke the cancel sub-protocol.

When \( A \) initiates the cancel sub-protocol, \( A \) will receive a cancellation token \( S_T(\text{"cancelled"}, B''_{\text{cancelled}}, h_A) \) from the TTP that proves which set of recipients have cancelled the exchange. If \( A \) holds \( h_{B_i} \) and the cancellation token, \( A \) can present them to the arbiter to settle the dispute of receipt without interrogating \( B_i \).

- With \( h_{B_i} \), \( A \) can prove \( B_i \) received \( c \).
- With \( S_T(\text{"cancelled"}, B''_{\text{cancelled}}, h_A) \), \( A \) can prove \( B_i \) received \( K \) if \( B_i \notin B''_{\text{cancelled}} \).

Then, \( A \) can prove \( B_i \) received \( M = D_K(c) \).

I-3. Some redundancy exists

In FPH protocol, some critical elements were “omitted” in order to simplify the explanation. On the other hand, some redundancy exists.

In the finish sub-protocol, \( h_{B_i,T} \) is a signature generated by the recipient \( B_i \) and used as evidence that \( B_i \) has demanded the TTP’s intervention. However, this evidence does not play any role in dispute resolution. When settling a dispute of receipt, if the sender \( A \) presents evidence \( h_{B_i} \), \( B_i \) cannot deny receiving the message \( M \) unless \( B_i \) can show the cancellation token \( S_T(\text{"cancelled"}, h_{B_i}) \) issued by the TTP. \( B_i \) cannot deny receipt of \( M \) by simply claiming that if the TTP cannot demonstrate \( h_{B_i,T} \), then \( B_i \) did not initiate the finish sub-protocol to obtain the key \( K \). (In fact, \( B_i \) may have received \( K \) from \( A \) at Step 3 in the exchange sub-protocol.) Therefore, \( h_{B_i,T} \) can be omitted in the finish sub-protocol.

In the cancel sub-protocol, \( S_T(B''_{\text{finished}}) \) is a signature generated by the TTP to notify \( A \) that \( B_i \in B''_{\text{finished}} \) has initiated the finish sub-protocol. This message can also be omitted as \( A \) only cares \( B''_{\text{cancelled}} \) from the TTP rather than \( B''_{\text{finished}} \). (Any \( B_i \) in \( B'' \) but not in \( B''_{\text{cancelled}} \) should obtain \( K \) and thus \( M \) either from \( A \) or from the TTP.) Even if it is used for notifying \( A \) of the current status, its definition is flawed since it lacks the critical information (e.g., \( h_A \)) that is related to a protocol run thus could be replayed by an attacker.

3 A Modified FPH Protocol

Here we present a modified version of FPH protocol, which overcomes the flaws and weaknesses identified in the previous section. The modified parts are underscored and the redundant parts are removed.
3.1 Notation

- $B = P_B(B_1), P_B(B_2), \cdots, P_B(B_1, B_2, \cdots)$: a set of intended recipients selected by the sender $A$.
- $B' = P_B'(B'_1), P_B'(B'_2), \cdots$: a subset of $B$ that have replied message 2 in the exchange sub-protocol.
- $B''_\text{cancelled}$: a subset of $B''$ (in plaintext) with which $A$ wants to cancel the exchange.
- $B''_\text{finished}$: a subset of $B''$ (in plaintext) that have finished the exchange with the finish sub-protocol.
- $M$: certified message to be sent from $A$ to $B$.
- $K$: symmetric key selected by $A$.
- $c = E_K(M)$: ciphertext of message $M$, encrypted with key $K$.
- $k_T = P_T(K)$: key $K$ encrypted with the TTP’s public key.
- $k_B = P_B'(K), P_B''(K), \cdots$: ciphertext of key $K$ that only the recipients in $B'$ can decrypt it.
- $h_A = S_A(H(c), B, TTP, H(A, B, K), k_T)$: first part of evidence of origin for every recipient $B_i \in B$.
- $h_B = S_B(H(c), A, TTP, H(A, B, K), k_T)$: evidence of receipt for $A$.
- $k_A = S_A(K, B')$: second part of evidence of origin for $B_i \in B'$.
- $k_T' = S_T(K, B_i)$: alternative second part of evidence of origin for $B_i$.
- $h_A' = S_A(H(c), k_T, h_A, B'')$: evidence that $A$ has demanded the TTP’s intervention to cancel the exchange sub-protocol with $B_i \in B''$.

3.2 Protocol Description

The modified exchange sub-protocol is as follows.

1. $A \Rightarrow B : c, H(A, B, K), k_T, B, TTP, h_A$
2. $B_i \rightarrow A : h_{B_i}$
3. $A \Rightarrow B' : k_{B'}, B', k_A$

If $A$ did not receive message 2 from some of the recipients $B''$, $A$ may initiate the following modified cancel sub-protocol.

1’. $A \rightarrow TTP : H(c), H(A, B, K), k_T, B, h_A, P_T(B''), h_{AT}$
2’. $TTP :$
   - FOR (all $B_i \in B''$)
   - IF ($B_i \in B''_\text{finished}$) THEN retrieves $h_{B_i}$
   - ELSE appends $B_i$ into $B''_\text{cancelled}$
3’. $TTP \rightarrow A :$ all retrieved $h_{B_i}$, $B''_\text{cancelled}$, $S_T(\text{"cancelled"}, B''_\text{cancelled}, h_A)$

---

$P_T(B_1, B_2, \cdots)$ is used by the TTP to check whether $B_i \in B$ when $B_i$ initiates the finish sub-protocol.
There will be different results if \( A \) does not set \( B'' = B - B' \) in the cancel sub-protocol. It is OK if \( A \) sets \( B'' \cap B - B' \), i.e., cancels some \( B_i \) that even replied with \( h_B \). (A possible scenario is that a \( B_i \) replied after \( A \) initiated the cancel sub-protocol.) But it is harmful for \( A \) if \( A \) sets \( B'' \subset B - B' \). That means a \( B_i \) in \((B - B') - B''\) is able to receive \( K \) with the finish sub-protocol (to decrypt \( c \)) while \( A \) does not have \( h_B \) to prove \( B_i \) received \( M \).

If some recipient \( B_i \) did not receive message \( 3 \), \( B_i \) may initiate the following modified finish sub-protocol.

\[
2'. \quad B_i \rightarrow TTP: \quad H(c), H(A,B,K), k_T, B, h_A, A, h_B, \space\space h_{AT} \rightarrow TTP \quad h_{AT}, \frac{S_T(<\text{cancelled}, h_B, h_{AT})}{h_{AT}} \rightarrow B_i \quad \text{IF } (B_i \in B''_{\text{cancelled}})
\]

ELSE

\[
3'. \quad TTP \rightarrow B_i: \quad \frac{S_T(K, k_T')}{P_B(K, k_T')}, k_T' \rightarrow B_i \quad \text{APPENDS } B_i \text{ INTO } B''_{\text{finished},} \quad \text{AND STORES } h_{B_i} \}
\]

If \( B_i \) received message \( 3 \), \( B_i \) needs to check whether \( K \) in \( k_A \) matches \( H(A,B,K) \) in \( h_A \). If not, \( B_i \) knows something wrong and should also initiate the finish sub-protocol. Then the TTP will check whether \( H(A,B, P_T(k_T)) = H(A,B,K) \). If not, \( B_i \) will be notified of the error, and neither \( A \) nor \( B_i \) will be committed to each other on the message exchange.

### 3.3 Dispute Resolution

The process of dispute resolution is modified as follows. In the dispute of origin, \( B_i \) has to provide \( M, c, K, H(A,B,K), k_T, B, TTP, h_A \) and \( B', k_A \) (or \( k_A' \)) to an arbiter. The arbiter will check

\( \text{(O-1) if } h_A \text{ is } A\text{'s signature on } (H(c), B, TTP, H(A,B,K), k_T) \text{, and } B_i \in B \); \)
\( \text{(O-2) if } k_A \text{ is } A\text{'s signature on } (K, B') \text{ and } B_i \in B', \text{ or if } k_T' \text{ is the } TTP\text{'s signature on } (K, B_i); \)
\( \text{(O-3) if the decryption of } c \text{ (i.e., } D_K(c) \text{) is equal to } M; \)
\( \text{(O-4) if } K \text{ certified in } k_A \text{ or } k_T' \text{ matches } H(A,B,K) \text{ certified in } h_A. \)

\( B_i \) will win the dispute if all of the above checks are positive.

In the dispute of receipt, \( A \) has to provide an arbiter with \( M, c, K, H(A,B,K), k_T, TTP, h_B, \) and \( B, B''_{\text{cancelled}}, h_A, S_T(<\text{cancelled}, B'', \text{cancelled}, h_A) \) if \( A \) has. The arbiter will check

\( \text{(R-1) if } h_B \text{ is } B_i\text{'s signature on } (H(c), A, TTP, H(A,B,K), k_T); \)
\( \text{(R-2) if } k_T \text{ is the encryption of } K \text{ with the } TTP\text{'s public key; } \)
\( \text{(R-3) if the decryption of } c \text{ (i.e., } D_K(c) \text{) is equal to } M; \)
\( \text{(R-4) if } H(A,B,K) \text{ and } k_T \text{ certified in } h_B \text{ match, i.e., } \)
\( H(A,B, P_T(k_T)) = H(A,B,K); \)
\( \text{(R-5) if } S_T(<\text{cancelled}, B'', \text{cancelled}, h_A) \text{ is the } TTP\text{'s signature, and } \)
\( B_i \notin B''_{\text{cancelled}}. \)
A will win the dispute if all of the above checks are positive. If the first four checks are positive but A cannot present evidence \( S_T('cancelled', B'_{\text{cancelled}}, h_A) \), the arbiter must further interrogate \( B_i \). If \( B_i \) is unable to present evidence \( S_T('cancelled', h_{B_i}, h_{AT}) \), A also wins the dispute. Otherwise, A will lose the dispute.

4 Extensions of Micalli Protocol

Here we first give a brief description of Micalli’s optimistic protocol for two-party certified email [10], then we extend this protocol to a multi-party scenario with asynchronous timeliness.

4.1 Micalli Protocol

Micalli presented two optimistic protocols in [10] for certified electronic mail (CEM) and electronic contract signing (ECS), respectively. The protocols were filed as a US patent No 5666420 in 1997 [9]. While CEM protocol is secure, ECS protocol has a security flaw as pointed out in [3]. Here we only review CEM protocol. To have a unified presentation, we use the same notation defined in the previous sections.

Before sending the plaintext message \( M \) to the recipient \( B \), the sender \( A \) computes a secret \( Z \) protected with the TTP’s public encryption key as

\[
Z = P_T(A, B, P_B(M))
\]

To achieve timeliness, Micalli proposed a cut-off time solution, where \( A \) chooses a time \( t \), after which the TTP should not help \( B \) in the conclusion of the protocol. The exchange sub-protocol is as follows.

1. \( A \rightarrow B : t, Z, S_A(t, Z) \)
2. \( B \rightarrow A : S_B(Z) \)
3. \( A \rightarrow B : P_B(M) \)

Whenever \( B \) reaches Step 1 and verifies \( A \)’s signature, he must extract the cut-off time \( t \) and estimate whether he will have enough time to contact the TTP in case of \( A \)’s misbehavior or channel failure. \( t_D \) denotes the maximum possible time discrepancy \( B \) believes that may exist between his clock and that of the TTP. If \( B \) receives Step 1 at time \( t_B \) (i.e., \( B \)’s local time) such that \( t_B + t_D \) is greater than or equal to \( t \), then \( B \) halts; otherwise he proceeds to Step 2. After verifying \( B \)’s signature, \( A \) sends \( P_B(M) \) to \( B \) at Step 3, and \( B \) can decrypt it with his private key to get \( M \).

\[\text{For simplicity, we assume that messages are encrypted directly with a public-key algorithm. But, according to standard practice, we could first encrypt a big message conveniently with a symmetric (session) key, and then encrypt this symmetric key with a public-key algorithm.}\]
After replying at Step 2, if B does not get the message within a reasonable amount of time, or \( Z = P_T(A, B, P_B(M)) \) does not hold, B contacts the TTP with the following finish sub-protocol.

\[
2'. B \rightarrow TTP: t, Z, S_A(t, Z), S_B(Z) \\
\text{IF } (t_{TTP} < t) \{3'. TTP \rightarrow B: X \\
4'. TTP \rightarrow A: S_B(Z) \}
\]

In this sub-protocol, the TTP verifies whether B’s request arrives before A’s cut-off time and also whether both 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 an unknown string \( X \), it sends \( X \) to B and forwards B’s signature to A.

### 4.2 Extension to Asynchronous Timeliness

We believe that a cut-off time is not the best solution for a timeliness property. Thus, 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 protocol, even if B approximately calculates in each run the time to contact the TTP, there can be always a situation in which the TTP is inaccessible for a longer time. In such a case B will not get the expected message and it will be difficult to figure out who bears the responsibility for the breach of fairness.

We introduce a new cancel sub-protocol. In this way, if A does not receive message 2 in the exchange sub-protocol, A can abort it with the cancel sub-protocol at any time. On the other hand, if B does not receive message 3 in the exchange sub-protocol, B can resolve it at any time with the finish sub-protocol.

The revised exchange sub-protocol (and the only one needed in case both parties behave and no error occurs in the communication channel) is as follows.

1. \( A \rightarrow B: Z, S_A(Z) \)
2. \( B \rightarrow A: S_B(Z) \)
3. \( A \rightarrow B: P_B(M) \)

The revised finish sub-protocol is as follows, which will be requested by B under the same conditions as the original one.

\[
2'. B \rightarrow TTP: Z, S_B(Z) \\
\text{IF cancelled} 3'. TTP \rightarrow B: S_A(cancel, Z) \\
\text{ELSE} \{3'. TTP \rightarrow B: P_B(M) \\
4'. TTP: \text{stores } S_B(Z) \}
\]

When the TTP receives such a request, it first checks B’s signature on \( Z \). If valid, the TTP further decrypts \( Z \) and extracts the identities of sender and recipient of \( Z \). If B is the intended recipient of \( Z \) and the exchange has not been cancelled by A, the TTP marks the exchange status related to \( (A, B, Z) \) as resolved, sends \( P_B(M) \) to \( B \), and stores \( S_B(Z) \) (which will be collected by A
when $A$ initiates the cancel sub-protocol). If the exchange has been cancelled by $A$, the TTP forwards $S_A(\text{cancel}, Z)$ to $B$, and $B$ can use this evidence to prove that $A$ has cancelled the exchange.

The new cancel sub-protocol is as follows.

1'. $A \to TTP: Z, S_A(\text{cancel}, Z)$

\text{IF resolved 2'. $TTP \to A: S_B(Z)$}

ELSE (2'. $TTP \to A: \text{ack}$

3'. $TTP$: stores $S_A(\text{cancel}, Z)$}

When the TTP receives such a request from $A$, it first checks $A$’s signature. If valid, the TTP further decrypts $Z$ and extracts the identities of sender and recipient of $Z$. If $A$ is the sender of $Z$ and the exchange status related to $(A, B, Z)$ is marked as resolved, the TTP forwards $S_B(Z)$ to $A$. Otherwise, the TTP marks the exchange status related to $(A, B, Z)$ as cancelled, and acknowledges $A$ of cancellation.

Although in [10] 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 arbiter will evaluate the final outcome of the protocol based on the evidence provided by the users. Consequently, if $A$ denies having sent a message in a CEM protocol run, $B$ should provide $(M, P_B(M), Z, S_A(Z))$ and the arbiter settles that $A$ sent the message $M$ if

\begin{itemize}
  \item $Z = P_T(A, B, P_B(M))$ holds, where $A$ and $B$ are the sender and recipient of $Z$, respectively;
  \item $A$’s signature on $Z$ is valid.
\end{itemize}

Similarly, if $B$ denies having received a message in a CEM protocol run, $A$ should provide $(M, P_B(M), Z, S_B(Z))$ and the arbiter settles that $B$ received the message $M$ if

\begin{itemize}
  \item $Z = P_T(A, B, P_B(M))$ holds, where $A$ and $B$ are the sender and recipient of $Z$, respectively;
  \item $B$’s signature on $Z$ is valid;
  \item $B$ cannot provide $S_A(\text{cancel}, Z)$.
\end{itemize}

We assume a deterministic public encryption algorithm is used. Otherwise, $A$ cannot discard the random seeds if a non-deterministic public encryption algorithm (e.g., the ElGamal cryptosystem [5]) is used.

4.3 Further Extension to Multi-Party Scenario

Here we further extend Micali’s two-party CEM protocol to a multi-party scenario with asynchronous timeliness as well. Some additional notation in the protocol description is as follows.

\begin{itemize}
  \item $B$: a set of intended recipients selected by the sender $A$.
\end{itemize}
\(- B' : \) a subset of \(B\) that have replied message 2 in the exchange sub-protocol.

\(- B'' = B - B' : \) a subset of \(B\) with which \(A\) wants to cancel the exchange.

\(- B''_{\text{cancelled}} : \) a subset of \(B''\) with which the exchange has been cancelled by the TTP.

\(- B''_{\text{finished}} : \) a subset of \(B\) that have finished the exchange with the finish sub-protocol.

\(- M : \) certified message to be sent from \(A\) to \(B\).

\(- P_B(M) = P_{B_1}(M), P_{B_2}(M), ... : \) an encryption concatenation of \(M\) for group \(B\).

\(- Z = P_T(A, B, P_B(M)) : \) a secret \(Z\) protected with the TTP's public encryption key.

The extended exchange sub-protocol is as follows.

1. \(A \Rightarrow B : Z, S_A(Z)\)
2. \(B_i \rightarrow A : S_{B_i}(Z)\) where each \(B_i \in B\)
3. \(A \Rightarrow B' : P_{B'}(M)\)

If \(A\) did not receive message 2 from some of the recipients \(B''\), \(A\) may initiate the following extended cancel sub-protocol.

1'. \(A \rightarrow TTP : P_T(B''), Z, S_A(\text{cancel}, B'', Z)\)
2'. TTP: 
   FOR (all \(B_i \in B''\))
   IF (\(B_i \in B''_{\text{finished}}\)) THEN retrieves \(S_{B_i}(Z)\)
   ELSE appends \(B_i\) into \(B''_{\text{cancelled}}\)
3'. TTP \(\rightarrow A : \) all retrieved \(S_{B_i}(Z), B''_{\text{cancelled}}, S_T(B''_{\text{cancelled}}, Z)\)

When the TTP receives such a request, it first checks \(A\)'s signature. If valid, the TTP further decrypts \(Z\) and extracts the sender’s identity. If \(A\) is the sender of \(Z\), the TTP checks which entities in \(B''\) have previously resolved the protocol and retrieves the evidence of receipt of those entities. Then, the TTP generates an evidence of cancellation for the rest of entities and includes everything in a message destined to \(A\).

If some recipient \(B_i\) did not receive message 3, \(B_i\) may initiate the following extended finish sub-protocol.

2'. \(B_i \rightarrow TTP : Z, S_{B_i}(Z)\)
IF (\(B_i \in B''_{\text{cancelled}}\)) 3'. TTP \(\rightarrow B_i : B''_{\text{cancelled}}, S_T(B''_{\text{cancelled}}, Z)\)
ELSE 
   \(\{3'. TTP \rightarrow B_i : P_{B_i}(M)\}\)
   4'. TTP: appends \(B_i\) into \(B''_{\text{finished}}\), and stores \(S_{B_i}(Z)\)

When the TTP receives such a request, it first checks \(B_i\)'s signature on \(Z\). If valid, the TTP further decrypts \(Z\) and extracts the identities of sender and recipients of \(Z\). If \(B_i\) is one of the intended recipients of \(Z\) and the exchange

\(5\) An efficient implementation for a big message \(M\) could be \(P_B(M) = E_K(M), P_{B_1}(K), P_{B_2}(K), ...\)
with $B_i$ has not been cancelled by $A$, the TTP sends $P_B(M)$ to $B_i$, and stores $S_B(Z)$ (which will be forwarded to $A$ when $A$ initiates the cancel sub-protocol).

If the exchange has been cancelled by $A$, the TTP sends $S_T(B''\_cancelled, Z)$ to $B_i$, and $B_i$ can use this evidence to prove that $A$ has cancelled the exchange.

If $B_i$ denies having received $M$, $A$ can present $B, B''\_cancelled, M, P_B(M), P_B(M), Z, S_B(Z), S_T(B''\_cancelled, Z)$ and the arbiter settles that $B_i$ received the message $M$ if

- $Z = P_T(A, B, P_B(M))$ holds, where $B_i \in B$ and $P_B(M) \in P_B(M)$;
- $B_i$’s signature on $Z$ is valid;
- The TTP’s signature on $S_T(B''\_cancelled, Z)$ is valid, and $B_i \notin B''\_cancelled$.

$A$ will succeed on the dispute if all the above checks are positive. If the first two checks are positive, but $A$ cannot present evidence of cancellation, then the arbiter must further interrogate $B_i$. If $B_i$ cannot present $S_T(B''\_cancelled, Z)$ in which $B_i \in B''\_cancelled$, $A$ also wins the dispute. Otherwise, $B_i$ can repudiate having received the message $M$. Therefore, evidence provided by the TTP is self-contained, that is, the TTP need not be contacted in case a dispute arises regarding the occurrence of the cancel sub-protocol launched by $A$.

If $A$ denies having sent $M$ to $B_i$, a similar process can be applied to settle such a dispute.

5 Conclusion

Certified email is a value-added service to deliver important data over the Internet with guaranteed receipt for each successful delivery. Multi-party certified email is useful when the same message needs to be sent to a set of recipients. In this paper, we analyzed Ferrer-Gomila et. al’s multi-party certified email protocol and further presented a modified version that overcomes its security flaws and weaknesses without compromising efficiency of the original protocol. We also extended Micali’s two-party certified email protocol to a multi-party scenario with asynchronous timeliness.

Regarding our two multi-party certified email protocols presented in Sections 3.2 and 4.3, respectively,

- Both of them maintain fairness no matter what happens in the execution of a protocol run. If an exception (e.g., a network failure or a party’s misbehavior) occurs, any party can rectify a potential breach of fairness by contacting the TTP.
- Both of them achieve asynchronous timeliness, i.e., any party can end a protocol run (with the cancel sub-protocol for the sender or the finish sub-protocol for the recipients) at any time without breach of fairness.
- Both of them are optimized in the sense that only 3 steps are required to complete a protocol run in the normal case. This is a lower bound for a certified email protocol.
In the modified FPH protocol, the certified message is split into two parts in delivery: a secret key selected by the sender, and the ciphertext of the message generated with this key. In the extended Micali protocol, however, the certified message is not split in delivery.

Such a difference has its own advantage and disadvantage to the two protocols. In comparison with the modified FPH protocol, the extended Micali protocol has simpler evidence for storage and dispute resolution but imposes heavier overheads on the TTP.

The performance of the two protocols is similar in the normal case where only the exchange sub-protocol is executed. However, when an exception occurs and the TTP is invoked, the TTP's communication and computing overheads of the two protocols are different. In the modified FPH protocol, the TTP only needs to receive, process, and forward the secret key while in the extended Micali protocol, the TTP needs to receive, process, and forward the whole certified message. Such a difference will be more significant if the size of the certified message is very large.

Acknowledgement

This work has been partially funded by the Ministry of Science and Technology under the project TIC2003-08184-C02-01. Additionally, the second author has been funded by the Consejería de Innovación, Ciencia y Empresa (Junta de Andalucía) under the III Andalusian Research Plan.

References


