On the Security of a Multi-Party Certified Email Protocol

Jianying Zhou
Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
jyzhou@i2r.a-star.edu.sg

December 12, 2003

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 ISC'02, Ferrer-Gomila et. al. presented a multi-party certified email protocol [5]. It has two major features. A sender could notify multiple recipients of the same information while only those recipients who acknowledged are able to get the information. In addition, its exchange protocol is optimized, which has only three steps. In this paper, we demonstrate some flaws and weaknesses in that protocol and propose amendments. The modified protocol is robust against various attacks while preserving the features of the original protocol.

Keywords: certified email, non-repudiation, security protocol

1 Introduction

Certified electronic mail is a value-added service of ordinary email service, 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 [1, 2, 3, 8, 9, 10]. 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. Multi-party certified email protocols was first proposed by Markowitch and Kremer, using an on-line trusted third party [6], or an off-line trusted third party [7].

In ISC'02, Ferrer-Gomila et. al. presented a multi-party certified email protocol [5]. It has two major features. A sender could notify multiple recipients of the same information while only those recipients who acknowledged are able to get the information. In addition, its exchange protocol is optimized, which has only three steps. However, this protocol suffers from a number of serious security problems. The objective of this paper is to analyze these problems and propose amendments. The modified protocol is secure against various attacks while preserving the features of the original protocol.

The rest of the paper is organized as follows. In Section 2, we briefly review the original protocol. After that, we demonstrate four attacks in Section 3, and further point out three weaknesses in Section 4. In Section 5, we present a modified version to overcome those security flaws and weaknesses. We conclude the paper in Section 6.

2 FPH Protocol

A multi-party certified email protocol was presented in [5]. 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.

Here we refer to this protocol as FPH protocol, and give a brief description with the same notation used in the original paper.

- X, Y: concatenation of two messages X and Y.
- $H(X)$: a collusion-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 $P_U^-(X)$: asymmetric encryption and decryption of message X.
- $S_U(X)$: principal U's digital signature on message X.
- $U \to V$: X: 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 $B_i \in B$.
- $h_{B_i} = S_{B_i}(H(c), k_T)$: evidence of receipt for A.
- $k_A = S_A(K, B')$: second part of evidence of origin for $B_i \in B'$.
- $k'_T = S_T(K, B_i)$: 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 $B_i \in B''$.
- $h_{B_iT} = S_{B_i}(H(c), k_T, h_A, h_{B_i})$: 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.

$$\begin{array}{ll} 1.\ A \Rightarrow B: & c, k_T, B, h_A \\ 2.\ B_i \to A: & h_{B_i} \\ 3.\ A \Rightarrow B': & K, B', k_A \end{array}$$

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

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

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

$$
\begin{array}{lll}
 & 2'.\ B_i \rightarrow TTP: & H(c), k_T, B, h_A, h_{B_i}, h_{B_iT} \\
\text{IF } (B_i \in B''\_cancelled) & 3'.\ TTP \rightarrow B_i: & S_T(\text{"cancelled"}, h_{B_i}) \\
\text{ELSE} & \{3'.\ TTP \rightarrow B_i: & K, k'_T \\
 & 4'.\ TTP: & \text{appends } B_i \text{ into } B''\_finished, \text{ and stores } h_{B_i}\}
\end{array}
$$

Dispute of Origin

In the case of dispute of origin, a recipient B_i clams 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', k_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 can 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.

3 Attacks

3.1 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.

At the *exchange* sub-protocol, the sender A could terminate without sending message 3. Then, 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 $S_T(\text{"}cancelled\text{"}, h_{B_i})$ 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 $S_T(\text{"}cancelled\text{"}, h_{B_i})$.

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.

$$h_A = S_A(H(c), B, \underline{TTP}, k_T)$$
$$h_{B_i} = S_{B_i}(H(c), \underline{TTP}, k_T)$$

$$\begin{array}{ll} 1.\ A \Rightarrow B: & c, k_T, B, \underline{TTP}, h_A \\ 2.\ B_i \rightarrow A: & h_{B_i} \\ 3.\ A \Rightarrow B': & K, B', k_A \end{array}$$

If A cheats in message 1 by encrypting K with a public key of the $TTP1$ but indicating to B_i as the TTP, A will not be able to get the valid evidence of receipt. When A presents $M, c,\ k_{T1} = P_{T1}(K),\ h_{B_i} = S_{B_i}(H(c), \underline{TTP}, k_{T1}),\ K$ 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} [1], 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.

3.2 How can B verify evidence of origin along

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

At the *exchange* sub-protocol, the sender A may send $K1$ and $k_{A1} = S_A(K1, B')$ 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.

[1] 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.

Suppose a non-deterministic public encryption algorithm, e.g. the ElGamal cryptosystem [4] 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 $S_T(\text{"}cancelled\text{"}, h_{B_i})$ 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 decrypted with the key that is consistent in $H(K)$ and k_T. The *exchange* sub-protocol is further modified as follows.

$$h_A = S_A(H(c), B, TTP, \underline{H(K)}, k_T)$$
$$h_{B_i} = S_{B_i}(H(c), TTP, \underline{H(K)}, k_T)$$

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

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 by providing $k_{T1} = P_T(K1)$ and $h_A = S_A(H(c), B, TTP, \underline{H(K)}, k_{T1})$ at Step 1, B_i will reply with $h_{B_i} = S_{B_i}(H(c), TTP, \underline{H(K)}, k_{T1})$. 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 $S_T(\text{"}cancelled\text{"}, h_{B_i})$. 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.

3.3 How to stop B misusing evidence of origin

In FPH protocol, it 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 Section 3.2, 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.

3.4 How to prevent collusion among recipients

In FPH protocol, fairness is a major security requirement. However, fairness may not be preserved if an intended recipient intercepts message 3 without replying message 2. This problem could be solved if the session key K in message 3 is encrypted in transmission. However, the breach of fairness may still occur 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 sightly 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.

[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.

4 Weaknesses

4.1 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 $S_T(\text{"}cancelled\text{"}, h_{B_i})$, 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 $S_T(\text{"}cancelled\text{"}, h_{B_i}, h_{AT})$. If a dispute arise, B_i can (and 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.

4.2 B may not be involved in dispute of receipt

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

When A initiates the *cancel* sub-protocol, A will get $S_T(\text{"}cancelled\text{"}, B''\_cancelled, h_A)$ 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 $S_T(\text{"}cancelled\text{"}, B''\_cancelled, h_A)$, A can prove B_i received K if $B_i \notin B''\_cancelled$. Then, A can prove B_i received $M = D_K(c)$.

4.3 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_iT} 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 $S_T(\text{"}cancelled\text{"}, h_{B_i})$ issued by the TTP. B_i cannot deny receipt of M by simply claiming that if the TTP cannot demonstrate h_{B_iT}, then B_i did not initiate the *finish* sub-protocol to obtain the key K. However, B_i may have received K from A at Step 3 in the *exchange* sub-protocol? Therefore, h_{B_iT} can be omitted in the *finish* sub-protocol.

In the *cancel* sub-protocol, $S_T(B''\_finished)$ is a signature generated by the TTP to prove that $B_i \in B''\_finished$ initiated the *finish* sub-protocol. Obviously, $B''\_finished$ can be derived easily as $B''\_finished = B'' - B''\_cancelled$. Therefore, $S_T(B''\_finished)$ is not required if the TTP includes B'' in $S_T(\text{"}cancelled\text{"}, B''\_cancelled, h_A)$. Moreover, if it is really needed, its definition is flawed since it lacks the critical information related to which protocol run thus could be replayed by an attacker.

5 Amendments

Here we present a complete modified version of FPH protocol, which overcomes the flaws and weaknesses identified in the earlier sections. The modified notification is as follows.

- $B = P_{B_1}(B_1), P_{B_2}(B_2), \cdots$: a set of intended recipients selected by the sender A. Each recipient's identity is encrypted with their own public key.
- $B' = P_{B'_1}(B'_1), P_{B'_2}(B'_2), \cdots$: a subset of B that have replied message 2 in the *exchange* sub-protocol.
- $B'' = B - B'$: a subset of B (in *plaintext*) with which A wants to cancel the exchange.
- $B''\_finished$: 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 TTP.
- 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.
- $k_{B'} = P_{B'_1}(K), P_{B'_2}(K), \cdots$: ciphertext of key K that only the recipients in B' can decrypt it.
- $h_A = S_A(H(c), B, TTP, H(K), k_T)$: first part of evidence of origin for every recipient $B_i \in B$.
- $h_{B_i} = S_{B_i}(H(c), TTP, H(K), k_T)$: 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.
- $h_{AT} = S_A(H(c), k_T, h_A, B'')$: evidence that A has demanded the TTP's intervention to cancel the *exchange* sub-protocol with $B_i \in B''$.

The modified *exchange* sub-protocol is as follows.

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

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

$$\begin{array}{ll} 1'.\ A \rightarrow TTP: & H(c), H(K), k_T, B, h_A, B'', h_{AT} \\ 2'.\ TTP: & \text{FOR (all } B_i \in B'') \\ & \text{IF } (B_i \in B''\_finished) \text{ THEN retrieves } h_{B_i} \\ & \text{ELSE appends } B_i \text{ into } B''\_cancelled \\ 3'.\ TTP \rightarrow A: & \text{all retrieved } h_{B_i},\ B''\_cancelled, \\ & S_T(\text{“cancelled”}, B'', B''\_cancelled, h_A) \end{array}$$

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

$$\begin{array}{lll} & 2'.\ B_i \rightarrow TTP: & H(c), H(K), k_T, B, h_A, h_{B_i} \\ \text{IF } (B_i \in B''\_cancelled) & 3'.\ TTP \rightarrow B_i: & B'', h_{AT}, S_T(\text{``cancelled''}, h_{B_i}, h_{AT}) \\ \text{ELSE} & \{3'.\ TTP \rightarrow B_i: & K, k'_T \\ & 4'.\ TTP: & \text{appends } B_i \text{ into } B''\_finished, \text{ and stores } h_{B_i}\} \end{array}$$

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

(O-1) if h_A is A's signature on $(H(c), B, TTP, H(K), k_T)$, 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;

(O-4) if K certified in k_A or k'_T matches $H(K)$ 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 $M, c, K, H(K), k_T, TTP, h_{B_i}$, and $B, B'', B''\_cancelled, h_A, S_T(\text{``cancelled''}, B'', B''\_cancelled, h_A)$ if A has. The arbiter will check

(R-1) if h_{B_i} is B_i's signature on $(H(c), TTP, H(K), k_T)$;

(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;

(R-4) if $H(K)$ and k_T certified in h_{B_i} match, i.e., $H(P_T^-(k_T)) = H(K)$;

(R-5) if $S_T(\text{``cancelled''}, B'', B''\_cancelled, h_A)$ is the TTP's signature, and $P_{B_i}(B_i) \in B$ but $B_i \notin B''\_cancelled$.

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 $S_T(\text{``cancelled''}, B'', B''\_cancelled, h_A)$, the arbiter must further interrogate $B_i$. If $B_i$ cannot present evidence $S_T(\text{``cancelled''}, h_{B_i}, h_{AT})$, A also wins the dispute. Otherwise, A will lose the dispute.

6 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. FPH multi-party certified email protocol is optimized in terms of the number of steps required in a protocol run. However, some problems exist in this protocol. In this paper, we identified several security flaws and weaknesses, and suggested how to overcome those problems. We further presented a modified protocol which is secure against various attacks without compromising efficiency of the original protocol. A formal analysis of this protocol is worthwhile to further check whether there are other subtle security problems.

References

[1] M. Abadi, N. Glew, B. Horne, and B. Pinkas. *Certified email with a light on-line trusted third party: Design and implementation*. Proceedings of 2002 International World Wide Web Conference, pages 387–395, Honolulu, Hawaii, May 2002.

[2] G. Ateniese, B. Medeiros, and M. Goodrich. *TRICERT: Distributed certified email schemes*. Proceedings of 2001 Network and Distributed System Security Symposium, San Diego, California, February 2001.

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

[4] T. ElGamal. *A public-key cryptosystem and a signature scheme based on discrete logarithms*. IEEE Transactions on Information Theory, IT-31(4):469–472, July 1985.

[5] J. Ferrer-Gomila, M. Payeras-Capella, and L. Huguet-Rotger. *A realistic protocol for multi-party certified electronic mail*. Lecture Notes in Computer Science 2433, Proceedings of 2002 Information Security Conference, pages 210–219, Sao Paulo, Brazil, September 2002.

[6] S. Kremer and O. Markowitch. *A multi-party non-repudiation protocol*. Proceedings of 15th IFIP International Information Security Conference, pages 271–280, Beijing, China, August 2000.

[7] O. Markowitch and S. Kremer. *A multi-party optimistic non-repudiation protocol*. Lecture Notes in Computer Science 2015, Proceedings of 3rd International Conference on Information Security and Cryptology, pages 109–122, Seoul, Korea, December 2000.

[8] M. Mut-Puigserver, J. Ferrer-Gomila, and L. Huguet-Rotger. *Certified electronic mail protocol resistant to a minority of malicious third parties*. Proceedings IEEE INFOCOM 2000, Volume 3, pages 1401–1405, Tel Aviv, Israel, March 2000.

[9] J. Zhou and D. Gollmann. *A fair non-repudiation protocol*. Proceedings of 1996 IEEE Symposium on Security and Privacy, pages 55–61, Oakland, California, May 1996.

[10] J. Zhou and D. Gollmann. *Certified electronic mail*. Lecture Notes in Computer Science 1146, Computer Security: Proceedings of 1996 European Symposium on Research in Computer Security, pages 160–171, Rome, September 1996.