Generic, Optimistic, and Efficient Schemes for Fair Certified Email Delivery

Guilin Wang[1], Feng Bao[1], Kenji Imamoto[2], and Kouichi Sakurai[2]

[1] Institute for Infocomm Research,
21 Heng Mui Keng Terrace, Singapore 119613
{glwang, baofeng}@i2r.a-star.edu.sg
[2] Kyushu University, Fukuoka, Japan
imamoto@itslab.csce.kyushu-u.ac.jp
sakurai@csce.kyushu-u.ac.jp

Abstract. As a value-added service for standard email systems, a certified email scheme allows a sender to deliver a message to a receiver in a *fair* way in the sense that either the sender obtains a receipt from the receiver and the receiver accesses the content of the email simultaneously, or neither party gets the expected item. In this paper, we first point out some weaknesses in several existing schemes. Then, we present two generic optimistic certified email schemes with transparent TTP. Our schemes are not only fair, but also support timeliness in two flavors: one scheme supports weak timeliness but with stateless TTP, while the other guarantees (strong) timeliness though only supports weak stateless TTP. Technical discussion and comparison are provide to show that our schemes are both secure and efficient, compared with the-state-of-art in this field.

Keywords: certified email, non-repudiation, fair exchange, security protocol.

1 Introduction

Certified email schemes are designed to achieve fair-exchange of a message and a receipt between two potentially mistrusting parties over the Internet. We call such a scheme is *fair*, if it is guaranteed that the message receiver can get the email content *if and only if* the message sender obtains an irrefutable receipt from the receiver. In other words, a dishonest party cannot obtain his/her expected item from a honest party in a cheating way such that the honest party is unable to get the corresponding item.

The undeniable receipt issued by the receiver serves as an evidence for non-repudiation of receipt (NRR). Namely, if the receiver denies having received the delivered message from the sender, the sender can provide publicly verifiable NRR evidence to an arbitrator to show that this is untrue. Some email schemes also provide evidences for non-repudiation of origin (NRO). Similarly, NRO evidence protects the receiver from the sender's dishonest denial that she has not deliver a particular message to the receiver, though this is the fact. We remark

S. Qing et al. (Eds.): ICICS 2005, LNCS 3783, pp. 40–52, 2005.

that certified email schemes supporting NRO services are functionally equivalent to non-repudiation protocols.

Since direct fair-exchange between two parties is extremely inefficient on both aspects of computation and communication, realistic solutions for certified email delivery need a trusted third party (TTP). Actually, according to the different usage of the TPP, certified email schemes (and non-repudiation protocols) can be classified in two types: schemes with on-line TTPs [8,16] or schemes with off-line TTPs [17,1,18,2,10,3,9,12,14]. Generally speaking, it would not be too difficult to carry out the first type schemes, due to the TTP's participation in every protocol instance (though maybe not in each step). However, those schemes are still expensive and inefficient in practice, since the TTP must offer services with high level of availability and performance, and the TTP is likely to become the system bottleneck if numerous certified emails per day have to be exchanged via the same TTP. A more appealing and promising approach is to exploit the TTP in an off-line fashion. Actually, those schemes with off-line TTPs are called as *optimistic* [1], since the TTP is not invoked in the protocol execution at all, unless one of the two parties misbehaves or the communication channel is out of order. In fact, most of researches in this area have focused on those optimistic fair-exchange protocols.

Most of certified email schemes [17,18,10,3,9,12,14] are designed to satisfy some or all of the following properties:

P1) **Optimism:** The TTP is involved only in abnormal cases, i.e., one party is trying to cheat or the communication channel fails to work.
P2) **Generic Construction:** Each party can *independently* exploit *any* (secure) standard signature scheme to generate non-repudiation evidences.
P3) **Transparent TTP:** The generated non-repudiation evidences are the same regardless of whether the TTP is involved in the protocol execution.
P4) **Stateless TTP:** To settle potential disputes between users, the TTP is not required to keep a database for remembering and searching the state information for each protocol instance.
P5) **High Performance:** To execute the protocol, both overheads of computation and communication could be reduced as much as possible.
P6) **NRR Service:** The protocol generates NRR (non-repudiation of receipt) evidence for the sender, which proves that the receiver received a specific message from the sender, if the receiver indeed obtained this message.
P7) **NRO Service:** The protocol generates NRO (non-repudiation of origin) evidence for the receiver, which proves that the sender delivered a specific message to the receiver, if the receiver indeed did this successfully.
P8) **Fairness:** After the completion of a protocol run, either each party obtains the expected item or neither party does.
P9) **Timeliness:** At any time during a protocol run, each party can *unilaterally* choose to terminate the protocol without losing fairness.
P10) **Confidentiality:** Except the receiver and the sender, the content of the delivered message cannot be accessed by anybody else, including the TTP.

In the above list, the first five properties are very meaningful in real-world systems to reduce the running cost of the TTP. As for the security requirements, fairness and NRR evidence are essential requirements for all certified email schemes, while NRO evidence, timeliness and confidentiality are optional. Actually, some schemes may only meet *weak* timeliness, i.e., one party can only terminate the protocol after waiting for a fixed amount of time, not on any moment during the protocol execution.

However, the existing schemes do not satisfy part of the above ten properties. In some cases, fairness, the most important security requirement for certified email, cannot be achieved due to the existence of some subtle but reasonable attacks [12]. This paper has two main contributions. We first identify some weaknesses in the certified email schemes proposed in [9,14]. After that, we present two certified email schemes supporting as many as possible properties. Specifically, one of our schemes supports weak timeliness but with stateless TTP, while the other satisfies (strong) timeliness though only supports weak stateless TTP. Except the difference on those two properties, all other eight properties are guaranteed by both schemes simultaneously. Table 1 shows that our schemes are not only secure but also efficient, when we compared them with the-state-of-art in this field.

Note that the on-line schemes in [8,16] are not enumerated in Table 1, since we only focus on optimistic protocols. In addition, to evaluate the efficiency we just compare the costs of both communication and computation in *normal* case, i.e., in the exchange protocols, since the abort and recovery protocols are only run rarely in abnormal cases. "Messages" means the number of message flows need to be transferred between the sender and receiver in each exchange protocol, while "Operations" denotes the number of asymmetric operations need to be performed by the two parties. Fairness in most of those schemes is affected by different potential attacks, so we mark those schemes with "Yes*" under the

Table 1. Comparison of Basic Features, Efficiency, and Security

	ZG [17]	ZDB [18]	GRV [12]	MK [13]	KM [10]	AN [3]	Micali [14]	IS [9]	**New I**	**New II**
P1. Optimism	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Yes**	**Yes**
P2. Generic Constr.	Yes	Yes	Yes	No	Yes	Yes*	Yes	Yes	**Yes**	**Yes**
P3. Trans. TTP	No	No	No	Yes	No	Yes	Yes	Yes	**Yes**	**Yes**
P4. Stateless TTP	No	No	No	No	No	Yes	Yes	Yes	**Yes**	**Weak**
P5. Messages	4	4	4	4	4	4	3	3	**3**	**3**
P5. Operations	8	12	10	12	8	17	8	8	**8**	**8**
P6. NRR Service	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Yes**	**Yes**
P7. NRO Service	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	**Yes**	**Yes**
P8. Fairness	Yes*	Yes*	Yes	Yes*	Yes*	Yes	Yes*	Yes*	**Yes**	**Yes**
P9. Timeliness	Weak	Yes	Yes	Yes	Yes	No	Weak	No	**Weak**	**Yes**
P10. Confidentiality	No	No	No	No	No	No	Yes	Yes	**Yes**	**Yes**

column of "fairness". Because those schemes are likely to be repaired by more or less modifications, though the security of revised schemes should be checked carefully again.

The rest of the paper is organized as follows. Section 2 introduces notations. In Section 3, we review and analyze the IS scheme [9]. After that, two new schemes are presented and then analyzed in Section 4 and Section 5, respectively. Finally, Section 6 concludes the paper.

2 Notations

In this paper, we use A, B, and T to denote unique identifiers of a sender Alice, a receiver Bob, and a TTP, respectively. m is a message Alice wants to deliver to Bob. $c = E_k(m)$ is the ciphertext of message m encrypted with a symmetric encryption algorithm $E_k(\cdot)$, where k is a session key selected by the sender Alice. $D_k(\cdot)$ denotes the corresponding symmetric decryption algorithm, i.e., we have $m = D_k(E_k(m))$. In addition, $H(\cdot)$ stands for a cryptographic hash function. $E_X(\cdot)$ means party X's asymmetric encryption algorithm, so that the resulting ciphertexts can only be decrypted by party X using its private key. $S_X(\cdot)$ denotes party X's secure signing algorithm, so that the resulting digital signatures can be verified by anybody using party X's signature verification key, which is assumed to be publicly available.

3 The IS Scheme and Its Security

In [9], the authors actually proposed two certified email schemes, one with on-line TTP, while the other with off-line TTP. Here, we only review and analyze their scheme with off-line TTP. For simplicity, we call it the IS scheme.

3.1 Review of the IS Scheme

(1) **Exchange Protocol.** When Alice wants to deliver a message m to Bob, they jointly executed the following exchange protocol:

$$\begin{array}{lll} \text{(e1).} & A \longrightarrow B: & A, T, c, EK, \mathsf{sub}, S_A(B, c, EK) \\ \text{(e2).} & B \longrightarrow A: & S_B(S_A(B, c, EK)) \\ \text{(e3).} & A \longrightarrow B: & E_B(k) \end{array}$$

The whole exchange protocol is explained in detail as follows.

1). The sender Alice first generates a random session key k, then computes $c = E_k(m)$, $EK = E_T(A, B, E_B(k))$, and her signature $S_A(B, c, EK)$. Then, Alice transmits $(A, T, c, EK, \mathsf{sub}, S_A(B, c, EK))$ to Bob, where sub is the subject of message m to encourage Bob receiving this encrypted email.

2). Upon receiving message flow (e1), Bob checks whether $S_A(B, c, EK)$ is Alice's valid signature on message (B, c, EK). If this is true and Bob would like to get this email from Alice, he returns back his signature $S_B(S_A(B, c, EK))$ to the sender Alice. Otherwise, Bob could ignore message flow (e1).

3). If Bob's signature $S_B(S_A(B, c, EK))$ is received timely and correctly, the sender Alice reveals $E_B(K)$ to the receiver Bob.
4). When $E_B(K)$ is received, the receiver Bob first derives the session key k by computing $k = D_B(E_B(k))$, and then obtains message $m = D_k(c)$.

(2) **Recovery Protocol.** If Bob honestly sent his receipt $S_B(S_A(B, c, EK))$ to Alice but Alice does not reveal $E_B(k)$ to him, Bob can initiate the following recovery protocol to get $E_B(k)$ from the TTP directly.

$$\begin{array}{lll} \text{(r1).} & B \longrightarrow T: & c, EK, S_B(S_A(B, c, EK)) \\ \text{(r2).} & T \longrightarrow B: & E_B(k) \\ & T \longrightarrow A: & S_B(S_A(B, c, EK)) \end{array}$$

Upon receiving a recovery request (r1), the TTP first decrypts EK to get $(A, B, E_B(k))$, and then checks whether $S_A(B, c, E_B(k))$ and $S_B(S_A(B, c, EK))$ are valid signatures. If both of them are correct, the TTP forwards $E_B(k)$ to Bob and $S_B(S_A(B, c, EK))$ to Alice.

(3) **Dispute Resolution Policy.** The original authors do not provide explicit dispute resolution policy, though they briefly mentioned that the generated signatures can be used as non-repudiation evidences (page 333 in [9]).

3.2 Security of the IS Scheme

In this section, we point out some weaknesses in the IS scheme.

(1) Protocol Specification. In the IS scheme, the protocol specification is not clear enough in the following senses. Firstly, when the exchange protocol is finished, Bob also needs to check whether $EK \equiv E_T(A, B, E_B(k))$. Otherwise, what he received may be an invalid NRO evidence. Secondly, the authors do not provide explicit definitions of NRO and NRR evidences, and exact procedures how a judge (or verifier) can validate those evidences. Obviously, just providing $S_B(S_A(B, c, EK))$ is not enough. Finally, there are no clear requirements on public encryption algorithms. That is, if those algorithms are randomized, how to deal with the random numbers utilized to produce $E_B(k)$ and EK. Simply requiring Alice should reveal those numbers to Bob may be not sufficient.

(2) An Attack. In the real world, there may be many TTPs. One user, say Bob, is probably not aware the identities of all TTPs. In such a scenario, a dishonest sender Alice can mount the following attack to disturb or even successfully cheat the receiver Bob.

0). Before executing the protocol, Alice and Bob have agreed to use a specific T as their TTP for certified e-mail delivery. However, Alice dishonestly exploits T''s public key to encrypt session key k by computing $EK' = E_{T'}(A, B, E_B(k))$, where T' is another TTP whose identity is unlikely known by Bob, for example in another city or country.
1). Then, Alice correctly prepares other values and sends the receiver Bob below message flow (e1'): $(A, T, c, EK', \mathsf{sub}, S_A(B, c, EK'))$.

2). According to the IS scheme, message flow (e1') is correct, so honest Bob will returns his signature $S_B(S_A(B, c, EK'))$ to Alice.
3). After that, however, Alice refuses to reveal $E_B(k)$ to Bob.
4). So Bob contacts the TTP T to get $E_B(k)$ by providing $c, EK', S_A(B, c, EK')$ and $S_B(S_A(B, c, EK'))$. But T will reject Bob's recovery request, since it cannot decrypt EK' correctly, due to the fact EK' is encrypted under the public key of T'.

The result of the above attack is that Bob would think the above protocol instance with Alice is unsuccessful (and then may delete related information sooner or later), but Alice has already gotten Bob's valid receipt, i.e., $(A, B, T', m, k, S_A(B, c, EK'), S_B(S_A(B, c, EK')))$. By showing this receipt, any judge will believe that Bob has received message m from Alice. It seems unreasonable to assume that Bob will contact with all (possible) TTPs for a recovery request. To avoid this attack, the TTP's identity T could be added in S_A and S_B.

We remark that a similar attack applies the schemes in [14].

(3) Efficiency. The IS scheme's efficiency could be improved in two aspects. (a) When Bob initiates the recovery protocol, he is required to send the TTP ciphertext c together other values. However, if m is a file with large size, such as digital movies, the communication overhead and delay are increased. (b) In the IS scheme, the non-repudiation evidence for both origin and receipt are the same, i.e., two embedded signatures $(S_A(B, c, EK), S_B(S_A(B, c, EK))$. So, to resolve a dispute the judge has to verify two signatures (and other values). In our new scheme, the judge only needs to validate one signature, and c will be replaced by $H(c)$ in our recovery protocol.

4 The Proposed Schemes

In this section, we present two new certified email schemes, which are secure and more efficient than the IS scheme. As well as the IS scheme, our schemes are generic optimistic protocols with transparent TTPs, and support fairness, confidentiality, both NRR and NRO evidences. Moreover, our schemes meet one more property: timeliness, which is not supported in the IS scheme. More specifically, one proposed protocol satisfies weak timeliness but with stateless TTP, while the other guarantees (strong) timeliness but the TTP needs to store some state information. Naturally, both versions could find their advantages in different environments.

To support the confidentiality of delivered message, we use the same idea in the IS scheme, i.e, $E_B(k)$ is embedded in EK. However, almost all ingredients and sub-protocols are re-designed. Of course, in this process we are inspired by the ideas appear in the literature, especially those in [6,12,9,14]. We assume that $E_B(\cdot)$ and $E_T(\cdot)$ are CCA2-secure randomized encryption algorithms. To emphasize a random number r is used to encrypted a message M, we write $C = E_T^r(M)$. Furthermore, we assume that both $E_B(\cdot)$ and $E_T(\cdot)$ have the property of *randomness recoverability*. Namely, in the procedure of decryption,

the owner of the secret decryption key can recover not only the message M but also the random number r used to produce the ciphertext. For simplicity, we denote this fact by $(M, r) = D_T(E_T^r(M))$. Actually, the OAEP series [15] are typical examples for such cryptosystems. As usual, the communication channel between Alice and Bob is assumed to be *unreliable*, i.e., messages inserted into such a channel may be lost. However, the TTP is linked with Alice and Bob by *reliable* communication channels, i.e., messages inserted into such a channel will be delivered to the recipient after a finite delay. In real life, such a communication channel could be implemented via computer networks compensated by other communication means, such as fax, telephone, or express mail service etc.

In addition, we introduce a new symbol P to denote the unique identifier for our protocol. P explicitly specifies the protocol name, version, and the cryptographic algorithms employed. By the appearance of P, related signatures can only be interpreted in a special way according to our definitions. We believe this is more logical and reasonable in practice. The following notations are re-defined or newly introduced.

- $\ell = H(P, A, B, T, HC, HK, t)$: A unique label to identify a protocol instance, where $H(\cdot)$ is a secure hash function. Label l means that a message m is supposed to be delivered from the sender Alice to the receiver Bob (with/without the TTP's help), where m is determined by a ciphertext c and a symmetric key k such that $m = D_k(c)$, $HC = H(c)$ and $HK = H(k)$. Here, t denotes a deadline with the meaning that after the expiration of t, the TTP does not accept a resolution request related to this t anymore. Label ℓ is used to identify a specific protocol instance, and link all messages generated in this instance.
- $EK = E_T^{r_2}(\ell, EB)$: Encrypted session key that includes label ℓ and $EB = E_B^{r_1}(k)$, where both r_1 and r_2 are random numbers. Compared with the IS scheme, both T and P are embedded in EK via label ℓ.
- $S_A = S_A(P, \ell, EK)$, and $S_B = S_B(P, \ell, EK)$: In contrast to the IS scheme, three changes are made here. (1) Non-repudiation evidences are defined as signatures on (P, ℓ, EK), not on (B, c, EK). However, note that since label ℓ is determined by $(P, A, B, T, H(c), H(k), t)$, so we have implicitly embedded the TTP's identifier T (and other information) in S_A and S_B. (2) Bob signs on message (P, ℓ, EK) directly, instead of signing on Alice's signature S_A. (3) To generate and verify label ℓ, the whole ciphertext c is not needed. So, in our recovery protocol, only $H(c)$ instead of c is submitted to the TTP. By doing so, the communication overhead between Bob and the TTP is further reduced.

4.1 Certified Email Scheme with Weak Timeliness

This new certified scheme consists of a exchange protocol, a recovery protocol, and a dispute resolution policy. Due to the absence of abort protocol, the TTP is not supposed to store any information related to a specific protocol instance. In addition, weak timeliness is achieved by the introduction of deadline t, which

could be negotiated by the two parties involved before the protocol executing. For example, let t be 24 hours after the beginning time of protocol execution.

Exchange Protocol. To send a message m, the sender Alice and receiver Bob execute the following exchange protocol collectively.

$$\begin{array}{lll} \text{(e1).} & A \longrightarrow B: & P, A, B, T, c, HK, EK, t, S_A(P, \ell, EK) \\ \text{(e2).} & B \longrightarrow A: & S_B(P, \ell, EK) \\ \text{(e3).} & A \longrightarrow B: & EB, r_2 \end{array}$$

That is, Alice first selects a session key k and two random numbers (r_1, r_2), and then computes the following values: $c = E_k(m)$, $HC = H(c)$, $HK = H(k)$, $\ell = H(P, A, B, T, HC, HK, t)$, $EB = E_B^{r_1}(k)$, $EK = E_T^{r_2}(\ell, EB)$, and $S_A(P, \ell, EK)$. After that, message flow (e1) is delivered to Bob. When Bob receives message flow (e1), he first recovers label ℓ and checks whether $S_A(P, \ell, EK)$ is indeed Alice's signature on (P, ℓ, EK). If this true, Bob further evaluates whether the included deadline t is sufficient for him to apply the TTP's help. If yes, Bob can return his signature $S_B(P, \ell, EK)$ to the sender Alice as message flow (e2). Otherwise, if any of the above checks fails or he does not want to receive this message from Alice, Bob can simply reject this email without any liability. If Bob's valid signature $S_B(P, \ell, EK)$ is received, Alice reveals (EB, r_2) to Bob. Finally, Bob checks whether $EK \equiv E_T^{r_2}(\ell, EB)$. If this true, Bob first decrypts (k, r_1) from EB, and then obtains the message m by computing $m = D_k(c)$. However, if Bob does not receive correct pair (EB, r_2) from Alice timely, he can execute the recovery protocol with the TTP (see below).

Recovery Protocol I. Whenever before the deadline t, Bob could initiate the following recovery protocol to get (EB, r_2) from the TTP directly.

$$\begin{array}{lll} \text{(r1).} & B \longrightarrow T: & P, A, B, T, HC, HK, EK, t, S_A(P, \ell, EK), S_B(P, \ell, EK) \\ \text{(r2).} & T \longrightarrow B: & \ell, EB, r_2 \\ & T \longrightarrow A: & \ell, S_B(P, \ell, EK) \end{array}$$

In detail, Bob first sends all related values to the TTP. Then, the TTP recovers label ℓ, and checks the validity of deadline t, $S_A(P, \ell, EK)$, and $S_B(P, \ell, EK)$. If everything is ok, it decrypts EK with its secret key. If the result is the expected pair (ℓ, EB) with a random number r_2, the TTP forwards (ℓ, EB, r_2) and $(\ell, S_B(P, \ell, EK)$ to Bob and Alice, respectively. However, if EK cannot be decrypted successfully, the TTP informs Bob that his recovery request is invalid.

Remark 1. Note that as we mentioned before, our recovery protocol has the following two features. First, in the theory the TTP is not required to store any information about a specific recovery request. The TTP's work is just to check the validity of a request, and then gives the corresponding answer. It does not need to remember anything except its decryption secret key. This is, this certified email scheme supports stateless TTP, though it only satisfies weak timeliness. Second, the overhead of communication between Bob and the TTP is independent of the size of message m, since only hash value $H(c)$, instead of c, is delivered to the TTP in the recovery protocol.

Remark 2. In the above scheme, Alice may need to wait Bob's signature S_B until the expiration of deadline t. That is, this certified email scheme only supports weak timeliness, since Alice cannot terminate a protocol run at *any* time. However, once Bob successfully executed the recovery protocol (before deadline t), Alice will receive the receipt from the TTP correctly. On the other hand, if Bob does not successfully apply recovery before the expiration of deadline t, this protocol run is deemed to be cancelled. In this situation, the result is still fair but unsuccessful, since neither Alice nor Bob can get their expected items.

Dispute Resolution Policy I.

- **Non-Repudiation of Origin.** To show that Alice indeed delivered message m to himself, Bob can provide $(P, A, B, T, m, k, r_1, r_2, t, S_A)$ to a judge. Then, the judge performs as follows:
  1) Compute $c = E_k(m)$, $EB = E_B^{r_1}(k)$, $\ell = H(P, A, B, T, H(c), H(k), t)$, and $EK = E_T^{r_2}(\ell, EB)$.
  2) Check whether S_A is Alice's valid signature on message (P, ℓ, EK). If yes, accept Bob's claim. Otherwise, reject Bob's claim.
- **Non-Repudiation of Receipt.** Similarly, to show that Bob has already received message m from herself, Alice can provide $(P, A, B, T, m, k, r_1, r_2, t, S_B)$ to a judge. Then, the judge performs as follows:
  1) Compute $c = E_k(m)$, $EB = E_B^{r_1}(k)$, $\ell = H(P, A, B, T, H(c), H(k), t)$, and $EK = E_T^{r_2}(\ell, EB)$.
  2) Check whether S_B is Bob's valid signature on message (P, ℓ, EK). If yes, accept Alice's claim. Otherwise, reject Alice's claim.

4.2 Certified Email Scheme with (Strong) Timeliness

In this version, the exchange protocol is the same as in the previous version. To support (strong) timeliness, we need to add an abort protocol, and modified the recovery protocol so that those two sub-protocols could work consistently.

Abort Protocol II. If Alice already delivered message flow (e1) to Bob but does not receive the expected $S_B(P, \ell, EK)$ correctly or timely, she can initiate the following abort protocol to cancel the protocol instance at *any* time before deadline t.

(a1). $A \longrightarrow T$: $P, A, B, T, HC, HK, EK, t, E_T^{r_3}(S_A(P, \ell, \mathsf{abort}))$
`if` request is invalid `then` stop
`if` (`state=aborted`) `then` retrieve $\ell, S_A(P, \ell, \mathsf{abort})$
$T \longrightarrow A$: $\ell, \mathsf{confirmation}$
`if` (`state=recovered`) `then` retrieve $\ell, S_B(P, \ell, EK)$
$T \longrightarrow A$: $\ell, S_B(P, \ell, EK)$
`else` set (`state=aborted`) and store $\ell, S_A(P, \ell, \mathsf{abort})$
(a2). $T \longrightarrow A$: $\ell, \mathsf{confirmation}$
$T \longrightarrow B$: $\ell, S_A(P, \ell, \mathsf{abort})$

To do so, Alice first sends message flow (a1) to the TTP, where $S_A(P, \ell, \mathsf{abort})$ serves as an *abort token* and is encrypted under the TTP's public key. When the TTP received such an abort request, it first recovers label ℓ, decrypts the last item, and then checks the validity of deadline t and signature $S_A(P, \ell, \mathsf{abort})$. If any of those checks fails, the TTP rejects Alice's request. Otherwise, it further checks whether label ℓ is already stored in its database. If yes, this means this protocol run identified by label ℓ has been aborted or recovered successfully, so the TTP retrieves corresponding items and forwards them to Alice. Otherwise, it sets state variable as `aborted`, stores $(\ell, S_A(P, \ell, \mathsf{abort}))$ into its database, forwards $(\ell, S_A(P, \ell, \mathsf{abort}))$ to Bob, and gives a confirmation to Alice. Here, confirmation can be defined as the TTP's signature on $(P, \ell, S_A(P, \ell, \mathsf{abort}))$.

Recovery Protocol II. Similarly, if Bob already sent $S_B(P, \ell, EK)$ to Alice but does not get correct pair (EB, r_2) from Alice in a reasonable period before deadline t, he can get this pair directly from the TTP by executing the following recovery protocol.

(r1). $B \longrightarrow T: \quad P, A, B, T, HC, HK, EK, t, S_A(P, \ell, EK), S_B(P, \ell, EK)$
`if` request is invalid `then` stop
`if` (`state=aborted`) `then` retrieve $\ell, S_A(P, \ell, \mathsf{abort})$
$T \longrightarrow B: \quad \ell, S_A(P, \ell, \mathsf{abort})$
`if` (`state=recovered`) `then` retrieve ℓ, EB, r_2
$T \longrightarrow B: \quad \ell, EB, r_2$
`else` set (`state=recovered`) and store $\ell, S_B(P, \ell, EK), EB, r_2$
(r2). $T \longrightarrow A: \quad \ell, S_B(P, \ell, EK)$
$T \longrightarrow B: \quad \ell, EB, r_2$

That is, Bob initially sends message flow (r1) to the TTP. Upon receiving such a recovery request, the TTP first recovers label ℓ, checks the deadline, verifies the signatures, and decrypts the ciphertext EK. If any of the above operations is unsuccessful, Bob's request will be rejected. Otherwise, the TTP further checks whether the protocol instance identified by label ℓ has been aborted or recovered successfully by searching its database. If yes, it retrieves corresponding items and forwards them to Bob. Otherwise, it sets state variable as `recovered`, stores $(\ell, S_B(P, \ell, EK), EB, r_2)$ into its database, forwards $(\ell, S_B(P, \ell, EK))$ and (ℓ, EB, r_2) to Alice and Bob, respectively.

Note that in the above scheme, timeliness is achieved since both Alice and Bob can terminate a protocol instance unilaterally at any moment before the deadline t. Different from other schemes supporting timeliness [18,10,13,12], however, deadline t is used in our scheme to support *weak* stateless TTP. That is, after time t the TTP could remove all state information related to deadline t into a log system. Therefore, the TTP only needs to maintain a relatively small searching database.

Dispute Resolution Policy II. This policy is almost the same as the Dispute Resolution Policy I, except the following difference. When the judge deals with non-repudiation of receipt, it further needs to inquire Bob or the TTP whether they could provide abort token $S_A(P, \ell, \mathsf{abort})$. If this is true, the

judge dismisses Alice's request. On the other hand, if all evidences are correct and no valid abort token is provided, the judge accepts Alice's claim.

Note that, we need to make the above change in our Dispute Resolution Policy II. Otherwise, Alice could cheat Bob as follows. Upon receiving Bob's signature S_B, Alice does not reveal (EB, r_2) to Bob. At the same time, she aborts the protocol run by contacting the TTP so that Bob cannot get (EB, r_2) from the TTP. According our new policy, however, in such case Bob can provide abort token to counteract the power of S_B so that the result is still fair for both parties. Moreover, in the abort protocol, it is also necessary to encrypt abort token $S_A(P, \ell, \mathsf{abort})$ under the TTP's public key. Otherwise, the following unfair situation may happen. After getting message flow (e1), Bob directly obtains (EB, r_2) from the TTP by successfully executing the recovery protocol. However, Alice may initiate the abort protocol almost simultaneously, since she does not receive Bob's signature. By eavesdropping the communication between Alice and the TTP, it is possible that Bob gets both (EB, r_2) and the abort token. So, Bob can access message m by using (EB, r_2), and deny that he already obtained m by providing abort token.

Remark 3. For simplicity, the same symbol P is used to denote the two protocol identifiers in the above two schemes. However, according our assumption, protocol identifier is unique for each protocol, so we actually need to differentiate them by using two distinct symbols, e.g., P_1 and P_2.

5 Security Discussion

In this section, we only argue that our second certified email scheme guarantees the fairness, the most important security requirement for all fair-exchange protocols. There are two reasons for this arrangement: (1) It is easy to see that other properties are satisfied as we claimed in Table 1; and (2) Fairness is also guaranteed in our first scheme by a similar but simpler argument. Intuitively, our schemes support fairness since both $S_A(P, \ell, EK)$ and $S_B(P, \ell, EK)$ can be explained as valid NRO and NRR evidences *if and only if* all related values $(c, HC, HK, EB, EK, t$ etc.) are prepared correctly. We classify our discussion into two cases: (1) Alice is honest, but Bob is trying to cheat; and (2) Bob is honest, but Alice is trying to cheat.

Case 1: *Alice is honest, but Bob is trying to cheat.* Alice is assumed to be honest now, so message flow (e1) is correctly prepared. When Bob receives such a valid message flow (e1), he has to ask himself whether he wants to access the encrypted message. If not, he could ignore this protocol run without loss fairness. If yes, Bob has to get (EB, r_2) at first. However, EB is securely encrypted under the TTP's public key, and prepared by Alice, so only Alice or the TTP can reveal the pair (EB, r_2) to Bob. Furthermore, according to the protocol specification, to get (EB, r_2) Bob is required to send his signature $S_B(P, \ell, EK)$ to Bob or the TTP before deadline t and under the condition that this protocol run is not aborted yet. In particular, this implies that Bob or anybody else cannot get (EB, r_2) by sending $S_B(P, \ell', EK)$ or $S_{B'}(P, \ell', EK)$ (and related information)

to the TTP. Because in such cases the TTP will find the first part of decrypted content of EK does not match label ℓ. Therefore, Bob cannot access the message or get valid NRO evidence without issuing receipt. That is, our scheme is fair for honest initiator (Alice).

Case 2: *Bob is honest, but Alice is trying to cheat.* The purpose of dishonest sender Alice is to get valid NRR evidence (i.e., $S_B(P, \ell, EK)$) from Bob so that Bob cannot access the message m or cannot get valid NRO evidence (i.e., $S_B(P, \ell, EK)$). However, this actually implies that Alice has to prepared message (e1) correctly. Otherwise, even if she got $S_B(P, \ell, EK)$ from Bob, this signature cannot be explained as valid NRR evidence, due to some inconsistency among m, c, HC, HK, EB, EK, t. The reason is that in our protocol, $S_B(P, \ell, EK)$ can be explained as valid NRR evidence showing that Bob indeed received message m from Alice, *if and only if* all the following conditions hold:

1) Alice can provide $(P, A, B, T, m, k, r_1, r_2, t, S_B(P, \ell, EK))$ such that $c = E_k(m)$, $EB = E_B^{r_1}(k)$, $\ell = H(P, A, B, T, H(c), H(k), t)$, $EK = E_T^{r_2}(\ell, EB)$, and $S_B(P, \ell, EK)$ is Bob's signature on (P, ℓ, EK).
2) Bob cannot provide valid abort token $S_A(P, \ell, \mathsf{abort})$.

Hence, the last cheating strategy for Alice is to thwart Bob getting the pair (EB, r_2) after she obtained Bob's receipt. Simply refusing to reveal (EB, r_2) to Bob is essentially unharmful to Bob, since the latter can execute the recovery protocol and then get this pair directly from the TTP before the deadline t. At the same time, as we mentioned after the description of our policy II, Alice also cannot achieve her goal by initiating the abort protocol as soon as she received Bob's signature $S_B(P, \ell, EK)$. The reason is that in this situation, the only two possible outputs are both fair: (a) This protocol run has been recovered, so Bob already received the (EB, r_2); or (b) This protocol will be aborted by the TTP, so Bob will get the abort token $S_A(P, \ell, \mathsf{abort})$, which is supposed to annul Bob's receipt $S_B(P, \ell, EK)$. Therefore, once again, our protocol does not enable the cheater (Alice) taking an advantage over the honest party (Bob). That is, our certified email scheme is fair for honest receiver.

6 Conclusion

In this paper, we first pointed out some weaknesses in several existing certified email schemes proposed in [9,14]. After that, we proposed two new generic optimistic schemes for certified email delivery. As Table 1 demonstrated, our protocols provide as many as possible desirable properties. At the same, the proposed solutions achieve better performance than the-state-of-the-art protocols in this field. Specifically, our schemes overcome the security shortcoming in schemes [9,14]; support transparent TTP while the schemes in [10,12] do not; provide evidences for non-repudiation of origin while the scheme in [3] does not; and satisfy timeliness while the schemes in [3,9] do not.

References

1. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In: *Proc. of AMC Conference on Computer and Communications Security* (*CCS'97*), pp. 7-17. ACM Press, 1997.
2. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18 (4): 591-606, 2000.
3. G. Ateniese and C. Nita-Rotaru. Stateless-receipient certified E-mail system based on verifiable encryption. In: *CT-RSA'02*, LNCS 2271, pp. 182-199. Springer-Verlag, 2002.
4. G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signature. In: *Proc. of AMC Conference on Computer and Communications Security* (*CCS'99*), pp. 138-146. ACM Press, 1999.
5. F. Bao, R.H. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line TTP. In: *Proc. of IEEE Symposium on Security and Privacy*, pp. 77-85. IEEE Computer Society, 1998.
6. F. Bao, G. Wang, J. Zhou, and H. Zhu. Analysis and improvement of Micali's fair contract signing protocol. In: *Information Security and Privacy* (*ACISP'04*), LNCS 3108, pp. 176-187. Springer-Verlag, 2004.
7. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: *Crypto'98*, LNCS 1462, pp.13-25. Springer-Verlag, 1998.
8. R. Deng, L. Gong, A. Lazar, and W. Wang. Practical protocol for certified electronic mail. *Journal of Network and Systems Management*, 1996, 4(3):279-297.
9. K. Imamoto and K. Sakurai. A cerified e-mail system with receiver's selective usage of delivery authortiy. In: *Indocrypt 2002*, LNCS 2551, pp. 326-338. Springer-Verlag, 2002.
10. S. Kremer and O. Markowitch. Selective receipt in cerified e-mail. In: *Indocrypt 2001*, LNCS 2247, pp. 136-148. Springer-Verlag, 2001.
11. S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25(17): 1606-1621. Elsevier, Nov. 2002.
12. S. Gürgens, C. Rudolph, and H. Vogt. On the security of fair non-repudiation protocols. In: *Information Security Conference* (*ISC'03*), LNCS 2851, pp. 193-207. Springer-Verlag, 2003.
13. O. Markowitch and S. Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In: *Information Security Conference* (*ISC'01*), LNCS 2200, pp. 363-378. Springer-Verlag, 2001.
14. S. Micali. Simple and fast optimistic protocols for fair electronic exchange. In: *Proc. of 22th Annual ACM Symp. on Principles of Distributed Computing* (*PODC'03*), pp. 12-19. ACM Press, 2003.
15. V. Shoup. OAEP reconsidered. *Journal of Cryptology*, 15(4): 223-249, 2002.
16. J. Zhou and D. Gollmann. Certified electronic mail. In: *Computer Security - ESORICS'96*, LNCS 1146, pp. 160-171. Springer-Verlag, 1996.
17. J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In: *Proc. of the 10th Computer Security Foundations Workshop* (*CSFW'97*), pp. 126-132. IEEE Computer Society Press, 1997.
18. J. Zhou, R. Deng, and F. Bao. Evolution of fair non-repudiation with TTP. In: *Information Security and Privacy* (*ACISP'99*), LNCS 1587, pp. 258-269. Springer-Verlag, 1999.