Extending A Key-Chain Based Certified Email Protocol with Transparent TTP

Zhiyuan Liu
University of Luxembourg &
Shandong University

Jun Pang
University of Luxembourg

Chenyi Zhang
University of Luxembourg &
University of New South Wales

***Abstract*—Cederquist et al. proposed an optimistic certified email protocol, which employs key chains to reduce the storage requirement of the trusted third party (TTP). We extend their protocol to satisfy the property of TTP transparency, using existing verifiably encrypted signature schemes. An implementation with the scheme based on bilinear pairing makes our extension one of the most efficient certified email protocols satisfying strong fairness, timeliness, and TTP transparency.**

I. Introduction

Certified email (CEM) protocols, as an extension of regular email services, require that both senders and receivers be responsible for their roles in the email services. That means, as a protocol successfully runs to the end, neither the sender can deny the dispatch of the email, nor can the receiver deny the receipt. Such requirements are usually implemented by a *non-repudiable evidence of origin* (EOO) that is to be acquired by the receiver, and a *non-repudiable evidence of receipt* (EOR) that is to be acquired by the sender. Both the EOO and the EOR may serve as evidences in case of a dispute, in order to prove the participation of the other party.

As a special class of fair exchange protocols [1], a CEM protocol is supposed to guarantee *fairness* with respect to non-repudiable evidences. Informally, at the end of a fair protocol run, either both parties acquire all the evidences, or no party gets an evidence. A *trusted third party* (TTP) might be introduced to take charge of the whole procedure and to provide undeniable records of submission (of the sender) and delivery (of the receiver). However in this way, a TTP may easily become a bottleneck, if she has to be involved in a large number of CEM services. A better solution, so called *optimistic* protocols [2], helps to release this burden from a TTP. In the optimistic protocols, a TTP is only required to be involved in case of unexpected events, such as a network failure or one party's misbehaviour, to restore fairness. In such situations, a TTP may digitally sign some pieces of information, which will be used later as evidences to guarantee that the protocol ends in a fair state. If both the signer and the receiver behave correctly and there is no presence of significant network delays, a CEM protocol terminates successfully without intervention of the TTP. A typical structure of an optimistic CEM protocol consists of an *exchange* sub-protocol, an *abort* sub-protocol and a *recover* sub-protocol. The exchange sub-protocol is executed by the communicating parties to deliver an email as well as exchanging undeniable evidences. The other sub-protocols are launched by a party to contact a TTP to deal with awry situations.

TTP *transparency* states that if a TTP has been contacted to help in a protocol, the resulting evidences will be the same as those obtained in the case where the TTP has not participated. In other words, by simply looking at the evidences, it is impossible to detect whether the TTP has been involved or not. Transparent TTPs are important and useful in practice, for instance, to avoid bad publicity. Besides, in many situations, an institution does not necessarily keep the up-to-date signatures or affidavits from all trusted services (especially when a TTP, who is trusted by the two parties involved in the protocol, may *not* be trusted by an external judge who is to verify the presented evidences). Moreover, this property also ensures privacy of the participants for asking for help from TTPs. In the context of CEM protocols, the use of a transparent TTP was first proposed by Micali [3], followed by a number of works [4], [5], [6], [7], [8], [9], [10], [11], [12], in which different cryptographic schemes are used to achieve TTP transparency, such as interactive proof of knowledge on the encrypted signatures, Schnorr-like signature schemes, and RSA-based encryption schemes.

In this paper, we focus on the development of a CEM protocol with a transparent TTP. Our starting point is the key-chain based protocol of Cederquist et al. [13]. The use of key chains is to reduce TTP's storage requirement. Our study exposes a weakness in the original protocol, for which we propose a fix. Later we extend Cederquist et al.'s protocol to satisfy the transparency of TTP, adopting a recently introduced verifiably encrypted signature scheme [14]. We are able to show, by a detailed comparison, that our protocol is one of the most efficient CEM protocols satisfying *TTP transparency*, in addition to the other important properties such as *strong fariness*, *effectiveness*, and *timeliness*.

Structure of the paper. We shortly explain security properties for CEM protocols in Sect. II. The CEM protocol using key chains is briefly described in Sect. III. Our extension with transparent TTP and its security analysis are detailed in Sect. IV. We compare our proposed protocol with some state-of-the-art CEM protocols supporting TTP transparency in Sect. V. We conclude the paper in Sect. VI.

II. Security Requirements

A CEM protocol needs to protect a participant who is *honest*, i.e., his behaviour strictly follows the protocol specifications. To this point, for the sake of readability, we write Alice for the sender and Bob for the receiver of an email. We assume the communication channels are *resilient*, in the sense that every message is guaranteed to reach its destination eventually. The following properties are typically required by an optimistic CEM protocol.[1]

Effectiveness. If no error occurs then the protocol successfully runs till the end without any intervention from TTP.

Timeliness. Both Alice and Bob have the ability to eventually finish the protocol anywhere during the protocol execution. This is to prevent endless waiting of an honest party in case of unexpectancies.

Fairness. Honest Alice (Bob) will get her (his) evidences, provided that the other party gets the evidence from her (him).[2] The evidences can be used to convince an adjudicator (who is not TTP) that Bob has received the mail, in Alice's case, or that Alice is the true sender of the message, in Bob's case. A protocol satisfies *strong fairness* if every judgement on Bob's (Alice's) non-repudiation can be made solely and independently from Alice's (Bob's) evidences, i.e., it does not necessarily involve TTP, nor the participation of Bob (Alice). If besides Alice's (Bob's) evidences, either TTP or Bob (Alice) needs to be contacted during the adjudication, the protocol only satisfies *weak fairness*.

TTP transparency. The evidence that each participant obtains is of the same format regardless of whether TTP is involved in the protocol execution or not.

III. A Certified Email Protocol using Key Chains

We describe the certified email protocol proposed by Cederquist et al. [13]. It makes use of key chains to reduce TTP's storage requirement. Once a key chain is initialized between two communication parties, the initiator can use any key within it to encrypt messages. Each exchange that uses the protocol is called a *protocol round*, and one initialisation phase followed by a number of protocol rounds is called a *protocol session*. Each protocol session belongs to a unique pair of communication parties. We focus on the main idea of the protocol, with its details available in the original paper [13].

We use $\{M\}^s_k$ to denote a message m encrypted with a symmetric key k, and $(M)_P$ to denote party P's signature on message M.[3]

A. Key chain generation

In optimistic CEM protocols, communicating parties will request TTP for help if the exchange process is disrupted. To achieve (strong) fairness, the TTP often needs to store sufficient amount of information, to have the ability to decrypt, retrieve or send out information for the protocol to finally reach a fair state. In most existing CEM protocols, the initiator uses either TTP's public key [7] or a separate key [9] to encrypt the email for each exchange. This first method normally requires asymmetric key operations, which are more expensive than symmetric key operations. The second method gives TTP burden of storing information of exchanges, such as involved parties, a hash value of email content and so on [15].

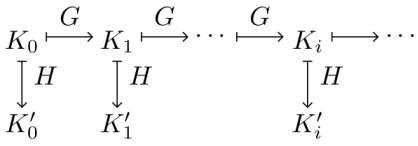

Fig. 1. A key chain.

To reduce the TTP's burden of storing too much information, the protocol [13] uses *key chains*. A chain of keys is a sequence of keys $K'_0, \ldots, K'_n$ (see Fig. 1), such that $K'_i := H(G^i(K_0))$ for each $i \geq 0$, where K_0 is the seed, $H : \kappa \to \kappa$ is a publicly known one-way collision-resistant hash function and $G : \kappa \to \kappa$ is a publicly known acyclic function (κ is a key domain). H and G functions are non-commutative, i.e., given an $H(K_i)$ for which K_i is unknown, it is infeasible to compute $H(G(K_i))$.

B. Initialisation

To initialise a session, the initiator Alice (A) sends the key chain seed K_0 and the identity of the potential responder Bob (B), together with a nonce nc to the TTP (T). TTP will check whether there already exists an entry $\langle A, B, K_0, \star\rangle$ in her database indicating whether the key chain has been established. If yes, TTP just ignores this request. Otherwise, TTP will choose a session identity sid, send a $\mathsf{cert} := (A, B, sid)_T$ to Alice, and store $\langle A, B, K_0, sid\rangle$ in her database.

C. Exchange sub-protocol

The i^{th} protocol round in a protocol session sid is described below. The round number i is initially 0 and can arbitrarily grow, Alice increments i after each round.

$$
\begin{array}{lll}
1^{ex}. & A \to B: & A, B, T, i, sid, h(K'_i), \{M\}^s_{K'_i}, \mathsf{EOO}_M, \mathsf{cert} \\
2^{ex}. & B \to A: & \mathsf{EOR}_M \\
3^{ex}. & A \to B: & K'_i \\
4^{ex}. & B \to A: & \mathsf{EOR}_{K'_i}
\end{array}
$$

where $\mathsf{EOO}_M := (B, T, i, sid, h(K'_i), \{M\}^s_{K'_i})_A$, $\mathsf{EOR}_M := (\mathsf{EOO}_M)_B$, $\mathsf{EOR}_{K'} := (A, K'_i, \{M\}^s_{K'_i})_B$ and h is just an ordinary hash function.

At first, Alice sends out message 1^{ex} to Bob. After receiving this, Bob checks the correctness of the signature on EOO_M and cert. If both are correct, Bob then commits himself to receiving the email by sending out message 2^{ex}. When Alice receives 2^{ex}, she checks the signature on EOR_M. If correct, Alice will send out K'_i to Bob. Upon receiving the key, Bob checks whether this key matches the hash value of the key that he received in message 1^{ex}. If yes, Bob decrypts the email

[1] There are more properties, such as *confidentiality*, *stateless TTP*, *accountability*, and *high performance*, which we do not discuss in this paper.

[2] Note that only honest participants need to be protected.

[3] In practice a signature is always applied on a hashed value.

and sends out a confirmation $\mathsf{EOR}_{K'}$ to indicate that he has received the key and the email.

D. Recovery sub-protocol

Both Alice and Bob have the right to run recovery sub-protocol by showing EOR_M. The recovery sub-protocol is mainly run with the aim of acquiring key K'_i or evidence $\mathsf{EOR}_{K'}$ with the help of TTP. Typically, Alice runs the recovery sub-protocol when she sends out key K'_i while not receiving message 4^{ex}, and Bob runs it when he sends out EOR_M while not receiving K'_i.

After receiving a recovery request from a party $p \in \{A, B\}$ of the form:

$$1^r. \quad P \to T: \quad f_r, A, B, h(K'_i), h(\{M\}^s_{K'_i}), i, sid, \mathsf{EOR}_M$$

where f_r is a flag used to identify the recovery request, TTP checks several things such as correctness of signatures, identities, entries for the key chain. If all checks succeed, TTP can retrieve K_0 and check whether $h(H(G^i(K_0)))$ matches $h(K'_i)$. If yes, TTP checks the status of round $status(i)$, indicating whether the exchange i has been resolved or aborted. Essentially, if $status(i)$ has not been set, TTP will set it as $h(\{M\}^s_{K'_i})$ and send back a *recovery token* $(A, B, h(\{M\}^s_{K'_i}), K'_i, i, sid)_T$ to the requester. If the round is aborted ($status(i) = \mathsf{a}$), TTP will send back an *abort token* $(A, B, h(\{M\}^s_{K'_i}), \perp, i, sid)_T$. If the status is different from $h(\{M\}^s_{K'_i})$ or any of the above tests fails, TTP will send back an *error* message in the form of $(\mathsf{error}, (\mathsf{error}, m^r)_T)$, where m^r is the content of the message in step 1^r. This error message indicates a misbehaviour and P can quit the protocol round.

E. Abort sub-protocol

Only Alice can abort, provided that the protocol has not yet been recovered. Typically, Alice aborts if she does not receive message 2^{ex}. To abort an exchange, Alice sends TTP the following message:

$$1^a. \quad A \to T: \quad f_a, A, B, i, sid, h(\{M\}^s_{K'_i}), \mathsf{abrt}$$

where f_a is a flag used to identify the abort request and abrt is Alice's signature on the abort request. After receiving this request, TTP checks several things such as correctness of signatures, identities, entries for the key chain, and $status(i)$ to make decisions. If $status(i)$ has not been initialised, TTP will set it as aborted ($status(i) := \mathsf{a}$) and send back an abort token. If the round is recovered, TTP checks whether $status(i) = h(\{M\}^s_{K'_i})$. If yes, TTP will send back a recovery token. Otherwise, an error message of the form $(\mathsf{error}, (\mathsf{error}, \mathsf{abrt})_T)$ is sent back.

F. Evidences and dispute resolution

When a disputation occurs, two parties can provide evidences to an external judge. For each protocol round i, EOO (evidence of origin) desired by Bob consists of

$$A, B, T, M, i, sid, K'_i, \mathsf{EOO}_M.$$

EOR (evidence of receipt) desired by Alice consists of

$$A, B, T, M, i, sid, K'_i, \mathsf{cert}, \mathsf{EOR}_M, \mathsf{EOR}_{K'}$$

if it is obtained by running the exchange sub-protocol. If Alice uses the recovery/abort sub-protocol, then EOR_M and $\mathsf{EOR}_{K'}$ will be replaced by the recovery token. In this case, EOR has the form of

$$A, B, T, M, i, sid, K'_i, \mathsf{cert}, (A, B, h(\{M\}^s_{K'_i}), K'_i, i, sid)_T.$$

As already remarked [13], the protocol is not TTP transparent, due to the fact that an observer can tell whether TTP was involved by simply checking EOR.

G. A vulnerability of the protocol

We found a vulnerability in the protocol. This vulnerability is mainly due to the form of $\mathsf{EOR}_{K'}$ that does not include any information about the current protocol round i. An $\mathsf{EOR}_{K'}$ in such form can be reused in different protocol rounds, which causes problems on fairness.

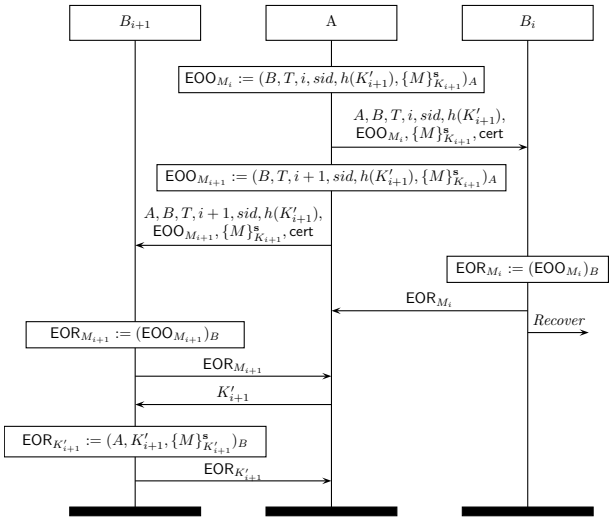

Fig. 2. A vulnerability on the CEM protocol using key chains.

Fig. 2 depicts a situation where dishonest Alice breaks strong fairness of honest Bob by reusing $\mathsf{EOR}_{K'}$. This attack requires multiple protocol rounds, which is sketched as follows. Alice first initiates an exchange i by sending out 1^{ex}, in which she uses K'_{i+1} instead of K'_i to encrypt the message, and gets the corresponding EOR_M, then does nothing for round i. Alice initiates another round $i + 1$ with Bob, and behaves honestly in order to acquire correct EOR_M and $\mathsf{EOR}_{K'}$ for round $i + 1$. Alice does this just because $\mathsf{EOR}_{K'}$ in both rounds i and $i + 1$ have exactly the same form, which is $(A, K'_{i+1}, \{M\}^s_{K'_{i+1}})_B$. At this moment, Alice has acquired all the necessary evidences for round i, leaving Bob in an unfair state. If Bob initiates a recovery sub-protocol, TTP will send back nothing but an error message because of the mismatch between $h(K'_i)$ and $h(K'_{i+1})$. As a result, for round i, strong fairness is broken. To fix this problem, we revise $\mathsf{EOR}_{K'}$ to be of the form $(A, i, K'_i, \{M\}^s_{K'_i})_B$, which includes the protocol round i.

IV. Transparent Trusted Third Party

We present a solution to extend the protocol in the previous section to support transparency of TTP. Our approach requires the usage of a *verifiably encrypted signature scheme* to encode Bob's commitment to receive the email in message 2^{ex}.

Notations. We write $(M)_{B|T}$ for Bob's verifiably encrypted (partial) signature on M, by using the public key of TTP to encrypt Bob's signature on M. Everyone can verify that $(M)_{B|T}$ is authentic, but only TTP and Bob are able to extract the complete signature $(M)_B$ out of $(M)_{B|T}$.

Exchange sub-protocol. The modified exchange sub-protocol is as follows:

$$\begin{array}{lll} 1^{ex}. & A \to B: & A, B, T, i, sid, h(K_i'), \{M\}^s_{K_i'}, \mathsf{EOO}_M, \mathsf{cert} \\ 2^{ex}. & B \to A: & \mathsf{EOR}^{\frac{1}{2}}_M \\ 3^{ex}. & A \to B: & K_i' \\ 4^{ex}. & B \to A: & \mathsf{EOR}_M \end{array}$$

where $\mathsf{EOR}^{\frac{1}{2}}_M$ is $(\mathsf{EOO}_M)_{B|T}$. After receiving EOR_M, Bob sends out his partial signature on EOO_M to show his commitment to receive the email. If Alice further sends Bob the key K_i', Bob will deliver a full signature back to Alice as EOR.

Abort and recovery sub-protocols. Alice is allowed to abort provided that she has sent out message 1^{ex}, but has not received message 2^{ex} from Bob. Once honest Alice and Bob contact TTP, they are not allowed to continue the exchange sub-protocol any longer.

Alice is allowed to launch the recovery sub-protocol provided she has sent out message 3^{ex}, but has not received message 4^{ex}. Similarly, Bob can launch the recovery sub-protocol if he has sent out message 2^{ex}, but has not received message 3^{ex}. The first message of the recovery sub-protocol for Alice is

$$1^r_A.\ A \to T: f_r, A, B, h(K_i'), h(\{M\}^s_{K_i'}), i, sid, \mathsf{EOR}^{\frac{1}{2}}_M, \mathsf{EOO}_M$$

where f_r is a flag used to identify the recovery request. The first message of the recovery sub-protocol for Bob is

$$1^r_B.\ B \to T: f_r, A, B, h(K_i'), h(\{M\}^s_{K_i'}), i, sid, \mathsf{EOR}_M, \mathsf{EOO}_M$$

On receipt of a message for recovery, TTP needs to check (1) the correctness of (verifiably encrypted) signatures on EOO_M and EOR_M ($\mathsf{EOR}^{\frac{1}{2}}_M$), (2) the identity of TTP, and (3) whether there is an entry in its database matching $\langle A, B, \star, sid\rangle$. If all the above checks succeed, TTP will retrieve K_0 and (4) check whether $h(H(G^i(K_0)))$ matches $h(K_i')$. If yes, TTP will check $status(i)$ for round i.

- If $status(i)$ has not been initialised, TTP will set $status(i) := h(\{M\}^s_{K_i'})$. Whenever necessary TTP converts $\mathsf{EOR}^{\frac{1}{2}}_M$ into EOR_M. After that, TTP sends out the following messages.

$$\begin{array}{lll} 2^r. & T \to B: & K_i', (K_i')_T \\ 3^r. & T \to A: & \mathsf{EOR}_M \end{array}$$

- If $status(i) = h(\{M\}^s_{K_i'})$, then TTP performs step 2^r and step 3^r (again).
- If $status(i) = \mathsf{a}$, TTP sends out the abort token to the one that launched the protocol.

$$2^r.\quad T \to A(B): \mathsf{abrt}, (\mathsf{abrt})_T$$

If any of the tests (1), (2), (3) and (4) fails, TTP ignores the recovery request and sends back an error message.

$$2^r.\quad T \to A(B): \mathsf{error}, (\mathsf{error}, m^r)_T$$

where m^r is the whole message received in step 1^r_A or 1^r_B.

Evidences and dispute resolution. When a disputation occurs, two parties can provide evidences to an external judge. For each protocol round i, EOO (evidence of origin) desired by Bob consists of

$$A, B, T, M, i, sid, K_i', \mathsf{EOO}_M.$$

EOR (evidence of receipt) desired by Alice consists of

$$A, B, T, M, i, sid, K_i', \mathsf{cert}, \mathsf{EOR}_M.$$

A. Security Analysis

As a special feature, the key chain provides an opportunity for Alice and TTP to have a predefined infinite list of symmetric keys. We assume that K_0, the seed of the chain, is a secret between Alice and TTP during related protocol executions. We restrict our attention to a single protocol round. For multiple rounds a weakness related to key chains has been identified and fixed in Sect. III, but the same vulnerability does not apply in the revised protocol in which $\mathsf{EOR}_{K_i'}$ is no longer used. In the following we informally justify that the revised protocol satisfies the claimed security properties.

Non-repudiation and fairness. If in round i Alice possesses EOR_M, we need to show that Bob must receive $\{M\}^s_{K_i'}$, K_i' and EOO_M in the same round i. There are three cases.

1) Alice receives EOR_M from Bob in message 4^{ex}, i.e., only the exchange sub-protocol is initialised and it successfully runs to the end, during which Bob obtains EOO_M, $\{M\}^s_{K_i'}$ and K_i'.
2) Alice receives EOR_M from message 3^r by launching the recovery sub-protocol herself, i.e., sending out 1^r_A. Then Alice must possess $\mathsf{EOR}^{\frac{1}{2}}_M$ from Bob's message 2^{ex}, which means Bob must have received 1^{ex} and obtained both EOO_M and $\{M\}^s_{K_i'}$. Also in the recovery sub-protocol, TTP must have sent message 2^r from which Bob obtains K_i'.
3) Alice receives EOR_M from message 3^r by Bob launching the recovery sub-protocol. Then Bob must have received 1^{ex} and have shown EOO_M to TTP, i.e., Bob obtains both EOO_M and $\{M\}^s_{K_i'}$, and in message 2^r Bob receives K_i' from TTP.

Furthermore, in case of a dispute, EOR_M alone from Alice is sufficient to prove that Bob has received M.

If in round i Bob possesses M, EOO_M and K_i', we need to show that (1) Alice must receive EOR_M in the same round i, and (2) Alice is the true sender of M. We know Bob can only receive $\{M\}^s_{K_i'}$ and EOO_M from 1^{ex}. There are two cases.

1) Bob receives K_i' from Alice, then the exchange sub-protocol runs at least up to message 3^{ex}. Bob may send 4^{ex} to Alice which contains EOR_M. If Bob does not send out 4^{ex}, Alice can always get EOR_M from TTP by launching the recovery sub-protocol.
2) Bob receives K_i' from TTP in the recovery sub-protocol. No matter who lauched the recovery sub-protocol, Alice gets EOR_M in message 3^r (from TTP).

As to the authenticity of the message M, Bob is able to convince every third party that M is indeed from Alice by verifying Alice's signature on EOO_M, after extracting M from $\{M\}^s_{K_i'}$ with K_i. Note that K_i' is also verified as its hashed value is contained in EOO_M too. Since presenting $(M, \mathsf{EOO}_M, K_i')$ is sufficient for Bob to prove that M is originally from Alice, together with the above EOR_M case for Alice, the protocol satisfies *strong fairness*.

Effectiveness. Suppose both Alice and Bob are honest, so that they faithfully follow the protocol in round i, and no error occurs, e.g., there is no significant network delays. It is obvious that only the exchange sub-protocol is launched, and it will stop at a state in which Alice obtains EOR_M, and Bob obtains both M and EOO_M.

Timeliness. In round i Alice can always launch the abort sub-protocol after she sends out message 1^{ex}, so that TTP will send back either an abort token or EOR_M depending on whether a recovery message has already arrived at TTP or not. Bob can launch the resolve sub-protocol any time after he receives message 1^{ex} and will get either an abort token or K_i', depending on the communication between Alice and TTP. The resilient channels between TTP, Alice and Bob guarantees that the above procedures terminate in a timely manner.

Transparency of TTP. If the exchange sub-protocol successfully runs to the end, Alice's evidence is EOR_M, and Bob's evidences are M, EOO_M and K_i'.

- Suppose the recovery sub-protocol is launched by Alice, then Alice must have message 2^{ex} which contains $\mathsf{EOR}^{\frac{1}{2}}_M$, and Bob must have message 1^{ex} which contains $\{M\}^s_{K_i'}$ and EOO_M. If TTP successfully verifies EOO_M and $\mathsf{EOR}^{\frac{1}{2}}_M$, TTP will convert $\mathsf{EOR}^{\frac{1}{2}}_M$ into EOR_M and send it back to Alice. Consequently TTP sends K_i' to Bob. In this case both Alice and Bob have the same evidences as only the exchange sub-protocol is launched.
- Suppose the recovery sub-protocol is launched by Bob, then Bob must have message 1^{ex}. If TTP successfully verifies EOO_M and EOR_M, TTP will forward EOR_M to Alice, and send K_i' to Bob, so that they get the same evidences in this case too.

In the next section, we discuss a particular signature scheme and our motivation to implement it in our protocol.

B. Verifiably Encrypted Signature Schemes

There is a variety of fair exchange protocols with verifiably encrypted signatures (sometimes also called convertible signatures) existing in the literature. Some of the earliest, such as Asokan et al. [16], Bao et al. [17], Boyed and Foo [18], and Camenisch and Damgård [19], apply interactive proof of knowledge on the encrypted signatures, such that more message exchanges are required in the protocols. Several later approaches use Schnorr-like signature schemes to wrap up signatures, such as by Ateniese and Nita-Rotaru [6], or RSA-based encryption schemes, such as by Markowitch and Kremer [4]. The GPS+RSA scheme used in [4] has been shown an attack by Cathalo et al. [20]. Another attack on fairness proposed by Bao [21] is applicable on a few of the signature schemes discussed in [22]. More recently, pairing algorithms for solving the Decision Diffie-Hellman problem in Gap Diffie-Hellman groups have been introduced to generate verifiably encrypted signatures [23], [14]. We briefly sketch the scheme of [14] as follows.

Let G_1 be a cyclic additive group generated by P with prime order q, and G_2 be a cyclic multiplicative group with the same order q. Let $e : G_1 \times G_1 \rightarrow G_2$ be a pairing operation satisfying *bilinearity*, i.e., $e(aX, bY) = e(X, Y)^{ab}$ for all $X, Y \in G_1$ and $a, b \in Z_q$. The signature scheme is as follows. Suppose G_1, G_2, P, e, q and H are all publicly available, where $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ is a cryptographic hash function and λ is the (private) key length. A user sets up $x \in Z_p$ as his secret key, and $X = x \cdot P$ for the corresponding public key. A signature on message M is $S = \frac{1}{H(M)+x} \cdot P$. To verify the signature, one only needs to check if $e(H(M) \cdot P + X, S) = e(P, P)$, note that $e(H(M) \cdot P + x \cdot P, \frac{1}{H(M)+x} \cdot P) = e(P, P)^{(H(M)+x)\cdot\frac{1}{H(M)+x}}$, by bilinearity. To produce a verifiably encrypted signature, suppose TTP has private key $y \in Z_q$ and public key $Y = y \cdot P \in G_1$, the new signature on M becomes $S' = \frac{1}{H(M)+x} \cdot Y$, with TTP's public key replacing G_1's group generator P. To verify one only needs to check if $e(H(M) \cdot P + X, S') = e(P, Y)$. TTP is able to get the true signature S by computing $y^{-1} \cdot S'$, where y^{-1} is the inverse of TTP's private key y (in G_2). Note that when applying this scheme to our extended protocol, both $e(P, P)$ and $e(P, Y)$ can be precomputed, thus reduces computation cost for a whole session.

We choose the algorithm in [14] since it requires fewer pairing operations than that of [23]. Moreover, there exist efficient pairing algorithms that implements pairing operations on elliptic curve-based point groups consuming time comparable to that of the RSA signatures of the same security level. It was studied in [24] that one 256-bit (prime field) pairing operation takes about 15 million clock cycles on a Core 2 Duo processor, which is the most expensive operation in the signature scheme.[4] According to [25], a 3072-bit RSA encryption (with a small exponent) takes about $620,000$ and a decryption takes about 28.6 million cycles on a Core 2 Duo processor.[5] In

[4]One 256-bit (prime field) pairing operation takes roughly 43 times as much as that of one point scalar multiplication [24], therefore, we ignore low cost operations such as point scalar multiplication and inverse operation in such signature schemes.

[5]If we implement the exponentiation algorithm by using Chinese Remainder Theorem, it will take roughly 22 million cycles with the `mpz_powm_sec()` function on a Core 2 Duo processor [25], which is still of comparable speed.

Protocol	Scheme	Fairness	Timeli.	$\#msg$	$\#op$
IS02 [5]	generic	strong*	No	3	4
Micali03 [7]	generic	strong*	weak	3	4
Wang06 [9]	generic	weak	Yes	3	2
MK01 [4]	RSA-based	strong*	Yes	4	7
AN02 [6]	RSA-based	strong	No	4	7
NZB04 [8]	RSA-based	strong*	No	4	4
MLCL06 [10]	RSA-based	strong*	No	4	8
HL08 [12]	RSA-based	strong	No	4	6
LCLQ08 [11]	bilinear pair	strong	No	4	5
Our protocol	bilinear pair	strong	Yes	4	3

TABLE I
AN OVERVIEW OF CEM PROTOCOLS SATISFYING TTP TRANSPARENCY.

practice, 3072-bit RSA signatures are of comparable security strength to 256-bit pairing-based signatures.

V. RELATED WORK

Our protocol supports TTP transparency, i.e., on the completion of a protocol run, the final structure and contents of the evidences possessed by both parties do not reveal whether TTP has intervened in the protocol or not. There are a number of CEM protocols in the literature (e.g. [4], [5], [6], [7], [8], [9], [10], [11], [12]) that supports the transparency of TTP, as listed in Tab. I. The protocol presented in this paper is the only one that satisfies strong fairness, timeliness and TTP transparency with a relatively low cost, to our knowledge.

In the above table, if the correctness of a protocol does not depend on any class of signature schemes, we write down that the protocol is *generic*. In such cases, the particular signature scheme is irrelevant, and the usage of verifiably encrypted signature or convertible signature is not required. We use "strong*" to indicate that it is claimed in the paper that the protocol satisfies strong fairness, but there exist attacks in the literature showing that the claim is invalid. All the protocols in the table satisfy TTP transparency, but they differ on other security properties such as timeliness and fairness. We also make comparisons on the number of messages as well as the computational costs as required by the protocols. We write "$\#msg$" for the number of messages in the exchange sub-protocol and "$\#op$" for the amount of computation equivalent to the number of RSA signature operations, i.e., we interpret other cryptographic operations as the number of RSA signatures referring to the best existing algorithms in the literature.

A. Timeliness

Only three protocols support timeliness (those of Wang [9], Markowitch and Kremer [4] and ours). In most cases the lack of timeliness is due to the fact that Alice is not allowed to abort after the first message. This design may trap Alice in a handicapped state, waiting forever on Bob's reply, without any effective ways to escape. Micali's protocol [7] satisfies weak timeliness by using a *cut-off* time, which indicates a deadline moment to resolve in a protocol run, in order to prevent endless waiting. However, this might cause problems if Alice and Bob cannot correctly estimate time differences between their local clocks and TTP's clock. Furthermore, in a real situation such mechanisms might enforce Bob to contact TTP as early as possible instead of replying to Alice if Bob is keen to proceed the current run.

B. Fairness

All the protocols except Wang's satisfy (or claim to satisfy) strong fairness. Wang's protocol [9] is not strongly fair, since when Alice is presenting Bob's EOR from the second message, the adjudicator has to contact either TTP or Bob in order to confirm that Alice has not aborted in the current run. If Alice has successfully aborted before Bob launches the recovery sub-protocol, and Alice has received the second message, Bob will not be able to obtain the key if Alice refuses to send out the third message. Micali's protocol [7] and Imamoto and Sakurai's protocol [5] are vulnerable to replay attack if Bob colludes with an outsider [26]. Both protocols in the work of Markowitch and Kremer [4] are shown to be unfair by the work of Gürgens et. al [15], in the way that if Bob colludes with an outsider, he is able to gain access to the message M without sending EOR_M back to Alice, by recovering the other protocol run with the outsider.[6] Moreover, the GPS scheme used by the second protocol in [4] has been proved insecure by Cathalo et al. [20]. Nenadić et al.'s protocol applies a particular RSA based verifiable encryption scheme, which has been shown by Ma et al. [10] that Bob is able to send an invalid partial signature which is undetectable by Alice and which is not recoverable by TTP. However the protocol in [10] is also identified with a similar attack by Hwang and Lai [12]. So far there exist no attacks on the fairness of Ateniese's protocol [6], Hwang et al.'s protocol [12] and Liang et al.'s protocol [11].

C. Efficiency

We concentrate on the amount of computation that involves in the exchange sub-protocol. The overloads of the abort and recovery sub-protocols are not considered, since those events are supposed to occur rarely. We define *one operation* as one 3072-bit RSA signature operation. As to the RSA scheme, the most time-consuming operation is modular exponentiation, and the ratio of the time taken for a modular exponentiation operation to the time taken for a single modular multiplication is linearly proportional to the exponent's bit length [27]. Therefore, we ignore single modular multiplications and the less time consuming algorithms such as symmetric encryption/decryption and hashing in protocols. As to pairing operation, just as Sect. IV-B states, in practice one 256-bit pairing operation can be faster than generating one 3072-bit RSA signature. As a conservative estimation we assume that verifying one pairing-based signature, the most time-consuming operation, also takes *one operation*. We omit the time used to generate a pairing-based signature as well as that used to verify an RSA-based signature in the analysis. For generic protocols we assume the RSA 3072-bit signature is

[6]This attack does not work on our key-chain based approach, because every key-chain is uniquely associated to a pair of sender and receiver. Bob and the colluding party are unable to recover from TTP unless Alice is involved.

used. In practice, they may choose faster schemes such as those of 256-bit elliptic curve cryptography (ECC) signatures.

From Tab. I, we conclude that the first three generic schemes are the most efficient, since they only need three messages in the exchange sub-protocol and at most 4 operations in computation. Nevertheless, none of them achieves TTP transparency, strong fairness and timeliness at the same time. As to the other four RSA based protocols, the number of operations varies from 4 to 8. In our protocol, it takes time equivalent to only 3 operations, since only 3 pairings are required (Bob needs to verify Alice's signature in message 1^{ex}, Alice needs to verify Bob's signature in message 4^{ex} and Bob's encrypted signature in message 2^{ex}). A signing operation in pairing-based scheme takes negligible amount of time. From Tab. I, only our protocol achieves all the three desirable properties – strong fairness, TTP transparency and timeliness – with a relatively low cost.

VI. Conclusion

We have proposed a TTP transparent CEM protocol, as an extension of Cederquist et al.'s protocol using key chains. To achieve this, we used a verifiably encrypted signature scheme based on bilinear pairing. Comparing to the existing CEM protocols, ours is among the most efficient ones satisfying effectiveness, strong fairness, timeliness, and TTP transparency.

The security analysis of the protocol in the paper is carried out in an informal way. In fact, the vulnerability on the original key chain based protocol [13] was found by a model checking exercise in Mocha [28]. We have verified strong fairness and timeliness of our protocol. In the future, we will study how to specify TTP transparency and formally check it. Another possible future work is to prove that it is impossible for an optimistic three-message CEM protocol to satisfy both strong fairness and strong timeliness.

Acknowledgement. We want to thank Sjouke Mauw for discussions on CEM protocols and David Galindo for discussions on verifiably encrypted signature schemes. Especially, we thank Johann Großschädl for experimental results on comparing efficiency of pairing operations and RSA signature operations.

References

[1] J. A. Onieva, J. Zhou, and J. Lopez, "Multiparty nonrepudiation: A survey," *ACM Computing Surveys*, vol. 41, no. 1, pp. 1–43, 2008.

[2] N. Asokan, M. Waidner, and M. Schunter, "Optimistic protocols for fair exchange," in *Proc. 4th ACM Conference on Computer and Communications Security (CCS)*. ACM, 1997, pp. 7–17.

[3] S. Micali, "Certified email with invisible post offices," 1997, an invited presentation at the RSA'97 conference.

[4] O. Markowitch and S. Kremer, "An optimistic non-repudiation protocol with transparent trusted third party," in *Proc. 4th Conference on Information Security (ICISC)*, ser. LNCS, vol. 2200. Springer, 2001, pp. 363–378.

[5] K. Imamoto and K. Sakurai, "A certified e-mail system with receiver's selective usage of delivery authority," in *Proc. 3rd Conference on Cryptology in India (INDOCRYPT)*. Springer, 2002, pp. 326–338.

[6] G. Ateniese and C. Nita-Rotaru, "Stateless-recipient certified e-mail system based on verifiable encryption," in *Proc. The Cryptographer's Track at RSA Conference 2002 (CT-RSA)*, ser. LNCS, vol. 2271. Springer, 2002, pp. 182–199.

[7] S. Micali, "Simple and fast optimistic protocols for fair electronic exchange," in *Proc. 22th Annual Symposium on Principles of Distributed Computing (PODC)*. ACM, 2003, pp. 12–19.

[8] A. Nenadić, N. Zhang, and S. Barton, "Fair certified e-mail delivery," in *Proc. 19th ACM Symposium on Applied Computing (ACM-SAC)*. ACM, 2004, pp. 391–396.

[9] G. Wang, "Generic non-repudiation protocols supporting transparent off-line TTP," *Journal of Computer Security*, vol. 14, no. 5, pp. 441–467, 2006.

[10] C. Ma, S. Li, K. Chen, and S. Liu, "Analysis and improvement of fair certified e-mail delivery protocol," *Computer Standards & Interfaces*, vol. 28, no. 4, pp. 467–474, 2006.

[11] X. Liang, Z. Cao, R. Lu, and L. Qin, "Efficient and secure protocol in fair document exchange," *Computer Standards & Interfaces*, vol. 30, no. 4, pp. 167–176, 2008.

[12] R.-J. Hwang and C.-H. Lai, "Efficient and secure protocol in fair e-mail delivery," *WSEAS Transactions on Information Science and Applications*, vol. 5, pp. 1385–1394, 2008.

[13] J. Cederquist, M. Torabi Dashti, and S. Mauw, "A certified email protocol using key chains," in *Proc. 3rd Symposium on Security in Networks and Distributed Systems (SSNDS)*. IEEE, 2007, pp. 525–530.

[14] F. Zhang, R. Safavi-Naini, and W. Susilo, "Efficient verifiably encrypted signature and partially blind signature from bilinear pairings," in *Proc. 5th Conference on Cryptology in India (INDOCRYPT)*, ser. LNCS, vol. 2904. Springer, 2003, pp. 71–84.

[15] S. Gürgens, C. Rudolph, and H. Vogt, "On the security of fair non-repudiation protocols," *International Journal of Information Security*, vol. 4, no. 4, pp. 253–262, 2005.

[16] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *Proc. 6th Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, ser. LNCS, vol. 1403. Springer, 1998, pp. 591–606.

[17] F. Bao, R. H. Deng, and W. Mao, "Efficient and practical fair exchange protocols with off-line ttp," in *Proc. IEEE Symposium on Security and Privacy (S&P)*, 1998, pp. 77–85.

[18] C. Boyd and E. Foo, "Off-line fair payment protocols using convertible signatures," in *Proc. 4th Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, ser. LNCS, vol. 1514. Springer, 1998, pp. 271–285.

[19] J. Camenisch and I. Damgård, "Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes," in *Proc. 6th Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, ser. LNCS, vol. 1976. Springer, 2000, pp. 331–345.

[20] J. Cathalo, B. Libert, and J.-J. Quisquater, "Cryptanalysis of a verifiably committed signature scheme based on GPS and RSA," in *Proc. 4th Conference on Information Security (ICISC)*, ser. LNCS, vol. 3225. Springer, 2004, pp. 52–60.

[21] F. Bao, "Colluding attacks to a payment protocol and two signature exchange schemes," in *Proc. 10th Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, ser. LNCS, vol. 3329. Springer, 2004, pp. 137–144.

[22] G. Ateniese, "Efficient verifiable encryption (and fair exchange) of digital signatures," in *Proc. 6th ACM conference on Computer and Communications Security (CCS)*, 199, pp. 138–146.

[23] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. 22th Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, ser. LNCS, vol. 2656. Springer, 2003, pp. 416–432.

[24] P. Grabher, J. Großschädl, and D. Page, "On software parallel implementation of cryptographic pairings," in *Proc. 13th Workshop on Selected Areas in Cryptography (SAC)*, ser. LNCS, vol. 5381. Springer, 2009, pp. 35–50.

[25] J. Großschädl, "Personal communications," 2010.

[26] S. Gürgens and C. Rudolph, "Security analysis of (un-) fair non-repudiation protocols," in *Proc. 1st Conference on Formal Aspects of Security (FASec)*, ser. LNCS, vol. 2629. Springer, 2002, pp. 229–232.

[27] D. Boneh, "Twenty years of attacks on the RSA cryptosystem," *Notice of the American Mathematical Society*, vol. 46, no. 2, pp. 203–212, 1999.

[28] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Tasiran, "Mocha: Modularity in model checking," in *Proc. 10th Conference on Computer Aided Verification (CAV)*, ser. LNCS, vol. 1427. Springer, 1998, pp. 521–525.