Timed-Release Encryption with Pre-open Capability and Its Application to Certified E-mail System*

Yong Ho Hwang, Dae Hyun Yum**, and Pil Joong Lee

Dept. of Electronic and Electrical Eng., POSTECH, Pohang, Korea.
yhhwang@oberon.postech.ac.kr,{dhyum,pjl}@postech.ac.kr

Abstract. We propose timed-release encryption with pre-open capability. In this model, the sender chooses a release time to open the message and a release key to pre-open, and encrypts the message using them. The receiver can decrypt the message only after the release time. When the sender wants the message to be opened before the release time, he may publish the release key. Then, the receiver can decrypt the message from his private key and the release key before the release time. However, an adversary cannot extract any information at any time even with the release key. We formalize the security model and provide an efficient construction secure under the BDH assumption in the random oracle model. In addition, we discuss the application of our schemes to efficient fair exchange systems such as a certified e-mail system.

1 Introduction

Timed-Release Encryption (TRE) is to "*sending message into the future*". In TRE, the sender transmits the encrypted message to the receiver and wants it to be decrypted after the appointed time. The receiver cannot decrypt the encrypted message until the release time. In the real world, TRE has many applications such as sealed-bid auctions, electronic voting, and payment system. There are two techniques used to construct TRE. One is based on time-lock puzzles where the time to recover a message is given by the minimum computational cost and the other is where a trusted third party (called the *time server*) is used to release the encrypted message at an appointed time. In the time-lock puzzle-based TRE, the receiver should make the computational effort to solve the relative time problem, which takes some required time. A time server-based TRE allows that the time server acts to release the message at the appointed time only. In general, time-lock puzzle-based schemes require a lot of computation effort for decryption. On the other hand, time server-based schemes require

* This research was supported by University IT Research Center Project, the Brain Korea 21 Project, and grant No. R01-2005-000-10713-0 from the research program of KOSEF.

** On leave at New York University, NY, USA.

J. Zhou et al. (Eds.): ISC 2005, LNCS 3650, pp. 344–358, 2005.

interaction between the server and the users and moreover should guarantee security against malicious behavior of the server. In this paper, we concentrate on time server based TRE schemes. The early works on this subject can be found in [12,17,9] and non-interactive timed-release schemes between the time server and the users using the bilinear map were recently proposed [4,14].[1]

Our Contribution. In applications using TRE, the sender may want to change the release time after sending message and the receiver can request a change of the pre-appointed release time. However, TRE schemes cannot deal with this problem because the release time is fixed when a message is encrypted. Hence, we introduce timed-release encryption with pre-open capability (TRE-PC) in a non-interactive model. '*Pre-open*' means that the receiver can decrypt the ciphertext before the release time in the sender's discretion. In the encryption phase of TRE-PC, the sender selects a secret value for the release key to allow the pre-open of the message. In case that the sender does not transmit the release key, the receiver can decrypt the message only after the release time. However, if the release key is given to the receiver, he can decrypt the message before the release time. Note that the release key have no information on message and the adversary cannot decrypt the message at any time even with the release key.

We propose the TRE-PC schemes using the bilinear map which have comparable efficiency compared to ordinary TRE schemes with the same domain. In our schemes, the time server periodically issues a kind of timestamp without interacting with the users. Our TRE-PC schemes satisfy the following properties.

- When the sender publishes the release key, only legitimate receiver can decrypt the ciphertext.
- Otherwise, no one, including the receiver and the time server, can decrypt the ciphertext before the release time.
- After the release time, only legitimate receiver can decrypt the ciphertext.
- In the encryption and decryption phases, the time server does not interact with the sender or the receiver.

We formalize a security model for TRE-PC and provide security proofs for our schemes. In addition, we show that TRE-PC can be efficiently applied to protocols for fair exchange (e.g. communication-efficient certified e-mail).

2 Model for TRE-PC

2.1 Generic Model

In a non-interactive model, the time server publishes his public key and periodically issues a timestamp. In the encryption phase of TRE-PC, the sender selects

[1] Boneh and Franklin mentioned that their identity-based encryption schemes with the bilinear map can also be applied to TRE in [6].

the desired release time to open the message and the release key for pre-open, and encrypts a message using the time server's public key and the receiver's public key. The receiver stores the ciphertext until the release time. After the release time, he can decrypt the message using the timestamp on the release time. If the sender wants to pre-open the ciphertext, he publishes the release key, with which the receiver can decrypt the ciphertext before the release time.

A *Timed-Release Encryption with Pre-open Capability* (TRE-PC) consists of 6 poly-time algorithms, (Setup, $\mathsf{Ext}_{\mathrm{TS}}$, $\mathsf{Gen}_{\mathrm{PK}}$, Enc, $\mathsf{Gen}_{\mathrm{RK}}$, Dec) such that:

- Setup: the *setup algorithm* takes a security parameter 1^k and returns the master-key mk and $\mathtt{params}$ (system parameters). The master key is known only to the "Time Server (TS)" and $\mathtt{params}$ is published.
- $\mathsf{Ext}_{\mathrm{TS}}$: the *timestamp extraction algorithm* used by the time server takes as input $\mathtt{params}$, mk and a release time t, and outputs a timestamp TS_t. The time server publishes a timestamp TS_t at time t.
- $\mathsf{Gen}_{\mathrm{PK}}$: the *key generation algorithm* takes as input a security parameter 1^k and $\mathtt{params}$, and generates the public key pk and the secret key sk.
- Enc: the *encryption algorithm* used by the sender takes as input a message M, a release time t, a randomly-chosen secret value v to generate a release key and pk, and outputs a ciphertext C.
- $\mathsf{Gen}_{\mathrm{RK}}$: the *release key generation algorithm* used by the sender takes as input v and a release time t, and returns the release key rk.
- Dec: the *decryption algorithm* is divided into two cases. If rk is published by the sender before the release time t, the receiver runs $M \leftarrow \mathsf{Dec}_{\mathtt{params}}(C, rk, sk)$. Otherwise, $M \leftarrow \mathsf{Dec}_{\mathtt{params}}(C, TS_t, sk)$ after time t.

2.2 Adversarial Model

The security of TRE-PC is related to the adversary's ability. In this model, we consider two types of adversaries: an *outside* adversary without the receiver's secret key and an *inside* adversary with the receiver's secret key. An *outside* adversary models either a dishonest time server or an eavesdropper who tries to decrypt the legal receiver's ciphertext (before or after the release time). An *inside* adversary models a legal receiver who tries to decrypt the ciphertext before the release time without the release key.

Security against Outside Adversary. We define the semantic securities against a chosen plaintext attack and a chosen ciphertext attack, which are now standard notions of security for public key encryption [5].

Definition 1. *Let $\mathcal{A}$ be an outside adversary. We say that a TRE-PC scheme $\mathcal{E}$ is semantically secure against a chosen ciphertext attack (IND-TR-CCA$_{\mathrm{OS}}$) if no polynomially bounded $\mathcal{A}$ has non-negligible advantage in the following game.*

- **Setup:** The challenger takes a security parameter 1^k and runs Setup and $\mathsf{Gen}_{\mathrm{PK}}$. The public key pk and the system parameters $\mathtt{params}$ are given to $\mathcal{A}$, while the master key mk and the secret key sk are kept secret.

- **Phase 1:** $\mathcal{A}$ makes extraction and decryption queries.
  - Extraction Queries $\langle t_i \rangle$. The challenger runs the $\mathsf{Ext}_{\mathrm{TS}}$ algorithm and generates the timestamp TS_{t_i} which is then given to $\mathcal{A}$.
  - Decryption Queries $\langle t_i, C_i \rangle$. The challenger runs the Dec algorithm and responds the resulting plaintext to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ selects two equal length messages M_0, M_1 and a release time t. The challenger picks a random bit b and gives $C = \mathsf{Enc}(M_b, t, v, pk)$ to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ makes extraction and decryption queries.
  - Extraction Queries $\langle t_i \rangle$. The same as Phase I.
  - Decryption Queries $\langle t_i, C_i \rangle \neq \langle t, C \rangle$. The challenger runs the Dec algorithm and responds the resulting plaintext to $\mathcal{A}$.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$ and wins if $b = b'$.

We define the advantage of the adversary $\mathcal{A}$ against the scheme $\mathcal{E}$ as the function of the security parameter k: $\mathrm{Adv}_{\mathcal{E},\mathcal{A}}^{\mathrm{IND-TR-CCA_{OS}}}(k) = |Pr[b = b'] - \frac{1}{2}|$.

Definition 2. *Let $\mathcal{A}$ be an outside adversary. We say that a TRE-PC scheme $\mathcal{E}$ is semantically secure against a chosen plaintext attack (IND-TR-CPA$_{\mathrm{OS}}$) if no polynomially bounded $\mathcal{A}$ has non-negligible advantage* $\mathrm{Adv}_{\mathcal{E},\mathcal{A}}^{\mathrm{IND-TR-CPA_{OS}}}(k)$ *in the above game without making decryption queries.*

Security against Inside Adversary. An *inside* adversary models the receiver who tries to decrypt the ciphertext without the release key before the release time. We define the security against the inside adversary as following.

Definition 3. *Let $\mathcal{A}$ be an inside adversary. We say that a TRE-PC scheme $\mathcal{E}$ is semantically secure against a chosen ciphertext attack (IND-TR-CCA$_{IS}$) if no polynomially bounded $\mathcal{A}$ has non-negligible advantage in the IND-TR-CCA$_{\mathrm{IS}}$ game.*[2]

3 Bilinear Map

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of prime order q. We denote $\mathbb{G}_1$ as an additive group and $\mathbb{G}_2$ as a multiplicative group. An (admissible) bilinear map $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ should satisfy the following properties [6,10]:

1. **Bilinear**: We say that a map $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is bilinear if $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$.

[2] The IND-TR-CCA$_{\mathrm{IS}}$ game is similar to the IND-TR-CCA$_{\mathrm{OS}}$ game except (1) the secret key sk is given to the adversary $\mathcal{A}$ in the **Setup** phase and (2) the adversary $\mathcal{A}$ is not allowed to make an extraction query with the target time t. For brevity, we do not consider the decryption queries, as the decryption oracle can be simulated by $\mathcal{A}$ with the secret key sk and extraction queries.

2. **Non-degenerate**: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$. Observe that since $\mathbb{G}_1$, $\mathbb{G}_2$ are groups of prime order this implies that if P is a generator of $\mathbb{G}_1$ then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_2$.
3. **Computable**: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

Bilinear Diffie-Hellman Assumption. The security of our scheme is proved under the hardness of the Bilinear Diffie-Hellman (BDH) problem: for given P, aP, bP, $cP \in \mathbb{G}_1$, compute $\hat{e}(P,P)^{abc} \in \mathbb{G}_2$. An algorithm $\mathcal{A}$ is said to solve the BDH problem with an advantage of ϵ if $\mathrm{Adv}^{\mathrm{BDH}}_{\mathcal{G},\mathcal{A}} = Pr[\mathcal{A}(P, aP, bP, cP) = \hat{e}(P,P)^{abc}] \geq \epsilon$ where the probability is over the random choice of a, b, $c \in \mathbb{Z}_q^*$ and the random bits used by $\mathcal{A}$. We assume that there is no polynomial time A to solve the BDH problem with non-negligible probability.

4 Construction of TRE-PC

Before describing our TRE-PC schemes, we introduce a simple dual encryption scheme using the bilinear map. Let $(V, S) = (vP, sP)$ be published and v be the sender's secret value. The sender encrypts a message M by $C = M \oplus \hat{e}(S, Q)^v$. Then the receiver can obtain the message M from the ciphertext C in case that he knows sQ or vQ; $M = C \oplus \hat{e}(V, sQ)$ or $M = C \oplus \hat{e}(S, vQ)$. We construct the TRE-PC schemes using this simple technique.

4.1 Basic Scheme

We present an efficient TRE-PC scheme secure against IND-TR-CPA$_{\mathrm{OS}}$/CCA$_{\mathrm{IS}}$.

- Setup: Given a security parameter 1^k, the following parameters are generated; two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, a generator P of $\mathbb{G}_1$, and two cryptographic hash functions H_1: $\{0,1\}^* \rightarrow \mathbb{G}_1^*$, H_2: $\mathbb{G}_2 \rightarrow \{0,1\}^n$ for some n. The time server chooses his master key $s \in \mathbb{Z}_q$ and computes his public key $S = sP$. The message space and the ciphertext space are $\{0,1\}^n$ and $\mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^n$ respectively. Then $\mathtt{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, S, H_1, H_2 \rangle$ is published.
- $\mathsf{Ext}_{\mathrm{TS}}$: At time t, the time server computes $Q_t = H_1(t)$ and publishes $TS_t = sQ_t$.
- $\mathsf{Gen}_{\mathrm{PK}}$: A user's secret key x is selected in $\mathbb{Z}_q$ and the public key Y is computed by xP. The user keeps his secret key and publishes the public key.
- Enc: The sender decides a release time t and selects $v \in_R \mathbb{Z}_q^*$ to make a release key. He encrypts a message M with a random number $r \in_R \mathbb{Z}_q^*$ as follows.

$$C = \langle rP, vP, M \oplus H_2(g_t) \rangle \text{ where } g_t = \hat{e}(rY + vS, Q_t)$$

- $\mathsf{Gen}_{\mathrm{RK}}$: When the sender wants the ciphertext to be decrypted before the release time, he computes the release key $V_t = vQ_t$ and publishes it.

- Dec: At time t, the receiver obtains TS_t from the time server. Then he can recover a message M from the ciphertext $C = \langle U, V, W \rangle$ as follows.

$$M = W \oplus H_2(\hat{e}(U, xQ_t) \cdot \hat{e}(V, TS_t))$$

If the sender publishes the release key V_t before the release time, then the receiver obtains M from $C = \langle U, V, W \rangle$ as follows.

$$M = W \oplus H_2(\hat{e}(U, xQ_t) \cdot \hat{e}(V_t, S))$$

The correctness can be checked by the following equation: $g_t = \hat{e}(rY + vS, Q_t) = \hat{e}(rxP, Q_t) \cdot \hat{e}(vsP, Q_t) = \hat{e}(rxP, Q_t) \cdot \hat{e}(P, vsQ_t) = \hat{e}(U, xQ_t) \cdot \hat{e}(V, TS_t) = \hat{e}(U, xQ_t) \cdot \hat{e}(V_t, S)$

This scheme requires only one pairing operation in the Enc phase while it provides timed-release encryption with pre-open capability. While two pairing operations are needed for decryption, $\hat{e}(U, xQ_t)$ can be pre-computed before obtaining the *timestamp* or the release key. Therefore, the decryption can be completed by additional one pairing operation at the release time.

Security analysis. We show the IND-TR-CPA$_{\mathrm{OS}}$/CCA$_{\mathrm{IS}}$ security of the above TRE-PC scheme under the BDH assumption in the random oracle model.

Theorem 4. *Suppose the hash functions H_1, H_2 are random oracles. Let the above scheme be* BasicTREPC. *Then* BasicTREPC *is secure against IND-TR-CPA$_{OS}$ under the BDH assumption. Namely:*

$$\mathrm{Adv}^{\mathrm{IND-TR-CPA_{OS}}}_{\mathsf{BasicTREPC},\mathcal{A}}(k) \leq \frac{q_{h_1} q_{h_2}}{2} \cdot \mathrm{Adv}^{\mathrm{BDH}}_{\mathcal{G},\mathcal{B}}(k)$$

where q_{h_1} and q_{h_2} are the number of H_1-queries and H_2-queries respectively.

This proof can be shown by simulating H_1, H_2 oracles and the extraction oracle as in Theorem 7. However, we omit the proof because of page restriction.

Next, we consider an *inside* adversary. In the following theorem, we show that BasicTREPC is secure against IND-TR-CCA$_{\mathrm{IS}}$.

Theorem 5. BasicTREPC *is secure against IND-TR-CCA$_{IS}$ under the BDH assumption. Namely:* $\mathrm{Adv}^{\mathrm{IND-TR-CCA_{IS}}}_{\mathsf{BasicTREPC},\mathcal{A}}(k) \leq \frac{q_{h_1} q_{h_2}}{2} \cdot \mathrm{Adv}^{\mathrm{BDH}}_{\mathcal{G},\mathcal{B}}(k)$.

Proof . Let $\mathcal{A}$ be an *insider* adversary that breaks IND-TR-CCA security of BasicTREPC with probability ϵ within time t making q_{h_1} and q_{h_2} hash queries. We show how to construct an adversary $\mathcal{B}$ to solve the BDH problem using $\mathcal{A}$.

- **Setup:** The BDH challenger gives an adversary $\mathcal{B}$ the BDH parameters $\langle q, \mathbb{G}_1, \mathbb{G}_1, \hat{e} \rangle$ and an instance $\langle P, aP, bP, cP \rangle$ of the BDH problem. The adversary $\mathcal{B}$ picks a secret key x in $\mathbb{Z}_q^*$ and computes the public key $Y = xP$. Then, he gives the adversary $\mathcal{A}$ the system parameters of BasicTREPC `params` $= \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, S, H_1, H_2 \rangle$ where $S = aP$. $\mathcal{B}$ simulates random oracles H_1, H_2 as follows.

- H_1**-queries:** $\mathcal{A}$ queries the value of a time t_i to the random oracle H_1. To respond to these queries $\mathcal{B}$ maintains a list of tuples $\langle t_i, k_i, Q_i \rangle$ called the H_1^{list}. The adversary $\mathcal{B}$ picks a random j where $1 \leq j \leq q_{h_1}$ before responding to H_1-queries. If the query t_i is already queried to H_1, then he returns $Q_i = H_1(t_i)$ in H_1^{list}. Otherwise, he picks a random element k_i in $\mathbb{Z}_q$ and computes $Q_i = k_i P$. Then he adds $\langle t_i, k_i, Q_i \rangle$ to H_1^{list}. Note that $\mathcal{B}$ responds $Q_j = cP$ to the query t_j instead of computing $k_j P$.
- H_2**-queries:** At any time $\mathcal{A}$ may issue queries to the random oracle H_2. To respond to these queries $\mathcal{B}$ maintains a list of tuples $\langle g_i, h_i \rangle$ called the H_2^{list}. If the query g_i is already queried to H_2, then he returns $h_i = H_2(g_i)$ in H_2^{list}. Otherwise, he picks a random value $h_i \in \{0,1\}^n$ and adds $\langle g_i, h_i \rangle$ to H_2^{list}.

- **Phase 1:** The adversary $\mathcal{A}$ makes queries to the extraction oracle and the encryption oracle. The adversary $\mathcal{B}$ simulates the oracles by answering as follows.
  - **Extraction-queries:** $\mathcal{A}$ queries t_i to obtain a timestamp TS_i. To respond to theses queries, $\mathcal{B}$ maintains a list of tuples $\langle t_i, Q_i, TS_i \rangle$ called the Ex^{list}. If $t_i = t_j$, $\mathcal{B}$ reports a failure and aborts. If the query t_i is already queried, $\mathcal{B}$ returns TS_i in Ex^{list}. Otherwise, $\mathcal{B}$ obtains $\langle k_i, Q_i \rangle$ such that $H_1(t_i)$=Q_i running the H_1 oracle. Then $\mathcal{B}$ computes $TS_i = k_i S$ $(= k_i aP = aQ_i)$ in $\mathbb{G}_1$ and records $\langle t_i, Q_i, TS_i \rangle$. It is returned to $\mathcal{A}$.
- **Challenge:** An adversary $\mathcal{A}$ outputs two equal-length messages (M_0, M_1) and a target release time t_i. If $i \neq j$, $\mathcal{B}$ reports a failure and aborts. Otherwise, he picks a random string $R \in \{0,1\}^n$ and returns $C = \langle rP, bP, R \rangle$ to $\mathcal{A}$. By definition, the decryption of C is $R \oplus H_2(\hat{e}(rP, xQ_i)\hat{e}(bP, aQ_i))$.
- **Phase 2:** The adversary $\mathcal{A}$ makes extraction queries as in Phase 1
- **Guess:** When $\mathcal{A}$ outputs its guess bit b', $\mathcal{B}$ picks a random element g_i in H_2^{list} and outputs $\frac{g_i}{\hat{e}(rP, xcP)}$ as the solution to the given BDH instance. The correctness is shown in following equations; $\frac{g_i}{\hat{e}(rP,xcP)} = \frac{\hat{e}(rP,xQ_i)\hat{e}(bP,aQ_i)}{\hat{e}(rP,xcP)} = \frac{\hat{e}(rP,xcP)\hat{e}(bP,aQ_i)}{\hat{e}(rP,xcP)} = \hat{e}(bP, acP) = \hat{e}(P,P)^{abc}$

If the adversary $\mathcal{B}$ does not abort during the simulation, then the adversary $\mathcal{A}$'s view is identical to its view in the real attack.

Lemma 6. *The probability that the adversary $\mathcal{B}$ outputs the correct answer of the BDH problem is at least $2\epsilon/q_{h_2}$ if the simulation does not fail.*

Proof . Let $\mathcal{H}$ be the event that the adversary $\mathcal{A}$ queries g_i for the correct answer to the random oracle H_2. $\mathcal{B}$ can derive the correct answer of the BDH problem from g_i as follows; $\frac{g_i}{\hat{e}(vS,cP)} = \frac{\hat{e}(aP,bQ_i)\hat{e}(vP,sQ_i)}{\hat{e}(vS,cP)} = \frac{\hat{e}(aP,bcP)\hat{e}(vsP,Q_i)}{\hat{e}(vS,cP)} = \frac{\hat{e}(aP,bcP)\hat{e}(vS,cP)}{\hat{e}(vS,cP)} = \hat{e}(P,P)^{abc}$. In the real attack, $Pr[b = b' | \neg\mathcal{H}] = 1/2$ because the decryption of C is independent to A's view if A did not query the correct g_i. In addition, the advantage of $\mathcal{A}$ in the real attack is $|Pr[b = b'] - 1/2| \geq \epsilon$. Therefore, $Pr[\mathcal{H}] \geq 2\epsilon$ is deduced as follows.

$$Pr[b = b'] = Pr[b = b'|\neg\mathcal{H}]Pr[\neg\mathcal{H}] + Pr[b = b'|\mathcal{H}]Pr[\mathcal{H}]$$
$$\leq Pr[b = b'|\neg\mathcal{H}]Pr[\neg\mathcal{H}] + Pr[\mathcal{H}] = \tfrac{1}{2}Pr[\neg\mathcal{H}] + Pr[\mathcal{H}] \leq \tfrac{1}{2} + \tfrac{1}{2}Pr[\mathcal{H}] \quad (1)$$
$$Pr[b = b'] \geq= Pr[b = b'|\neg\mathcal{H}]Pr[\neg\mathcal{H}] = \tfrac{1}{2}Pr[\neg\mathcal{H}] = \tfrac{1}{2} - \tfrac{1}{2}Pr[\mathcal{H}] \quad (2)$$

Then we have $Pr[\mathcal{H}] \geq 2\epsilon$ from $\epsilon \leq |Pr[b = b'] - \frac{1}{2}| \leq \frac{1}{2}Pr[\mathcal{H}]$ by (1), (2). By the way, the adversary $\mathcal{A}$ simulated by the adversary $\mathcal{B}$ does not distinguish the real environment and the simulated environment. Therefore, $Pr[\mathcal{H}]$ in the real attack is the same as $Pr[\mathcal{H}]$ in the simulation. When the event $\mathcal{H}$ happens, the probability that $\mathcal{B}$ chooses the correct query in H_2^{list} is $1/q_{h_2}$. In consequence, $\mathcal{B}$ has the probability of at least $2\epsilon/q_{h_2}$. □

Let q_{ex} be the number of the extraction queries. The probability that the attack is failed in the extraction phase is $(1 - g_{ex}/q_{h_1})$ and that in the challenge phase is $1/(g_{h_1} - q_{ex})$. Therefore, the probability that $\mathcal{B}$ does not abort is $1/g_{h_1}$. In consequence, we can obtain our result; $\mathrm{Adv}^{\mathrm{IND-TR-CCA_{IS}}}_{\mathsf{BasicTREPC},\mathcal{A}_{IS}}(k) \leq \frac{q_{h_1}q_{h_2}}{2} \cdot \mathrm{Adv}^{\mathrm{BDH}}_{\mathcal{G},\mathcal{B}}(k)$ by Definition 3. □

4.2 TRE-PC Secure Against CCA

To provide IND-TR-CCA$_{\mathrm{OS,IS}}$ security, we modify our scheme with the technique of the REACT scheme proposed by Okamoto and Pointcheval [15].

- Setup: Given a security parameter 1^k, the following parameters are generated; two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, a generator P of $\mathbb{G}_1$, and three cryptographic hash functions H_1: $\{0,1\}^* \to \mathbb{G}_1^*$, H_2: $\mathbb{G}_2 \to \{0,1\}^n$ H_3: $\{0,1\}^* \to \{0,1\}^{k_2}$. The time server chooses the master key $s \in \mathbb{Z}_q$ and computes the public key $S = sP$. The message space and the ciphertext space are $\{0,1\}^n$ and $\mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \times \{0,1\}^n \times \{0,1\}^{k_2}$ respectively. Then $\mathtt{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, S, H_1, H_2, H_3\rangle$ is published.
- $\mathsf{Ext}_{\mathrm{TS}}$, $\mathsf{Gen}_{\mathrm{PK}}$: The same as the BasicTREPC scheme.
- Enc: The sender decides a release time t and selects $v \in_R \mathbb{Z}_q^*$ to make a release key. He encrypts a message M with random values $r \in_R \mathbb{Z}_q^*$ and $g_R \in \mathbb{G}_2$ as follows.

  $$C = \langle U, V, W, Z, \sigma\rangle = \langle rP, vP, g_R \cdot g_t, M \oplus H_2(g_R), H_3(g_R, M, U, V, W, Z)\rangle$$

  where $g_t = \hat{e}(vS + rY, Q_t)$.
- $\mathsf{Gen}_{\mathrm{RK}}$: The same as the BasicTREPC scheme.
- Dec: At time t, the receiver obtains TS_t from the time server. Then he can derive a message M from the ciphertext $C = \langle U, V, W, Z, \sigma\rangle$ as follows.

  $$M = Z \oplus H_2(W/(\hat{e}(U, xQ_t) \cdot \hat{e}(V, TS_t)))$$

  If the sender publishes the release key V_t before the release time, then the receiver obtains M from $C = \langle U, V, W, Z, \sigma\rangle$ as follows.

  $$M = Z \oplus H_2(W/(\hat{e}(U, xQ_t) \cdot \hat{e}(V_t, S)))$$

If $\sigma \neq \sigma'$ where $\sigma' = H_3(W/(\hat{e}(U, xQ_t) \cdot \hat{e}(V, TS_t)), M, U, V, W, Z)$ or $\sigma' = H_3(W/(\hat{e}(U, xQ_t) \cdot \hat{e}(V_t, S)), M, U, V, W, Z)$, then the receiver regards the ciphertext as invalid.

Security analysis. We show the IND-TR-CCA$_{\mathrm{OS,IS}}$ security of the above TREPC scheme under the BDH assumption in the random oracle model.

Theorem 7. *Suppose the hash functions H_1, H_2, H_3 are random oracle. Let the above scheme be* FullTREPC. *Then* FullTREPC *is secure against IND-TR-CCA$_{OS}$ over the BDH assumption. Namely:*

$$\mathrm{Adv}^{\mathrm{IND-TR-CCA_{OS}}}_{\mathsf{FullTREPC},\mathcal{A}}(k) < \frac{g_{h_1} g_{h_2}}{2} \mathrm{Adv}^{\mathrm{BDH}}_{\mathcal{B}}(k) + \frac{q_d}{2^{k_2}}$$

where q_{h_1}, q_{h_2} and g_d are the number of H_1-queries, H_2-queries and decryption queries respectively.

Proof . Let $\mathcal{A}$ be an *outsider* adversary who breaks the IND-TR-CCA security of FullTREPC with probability ϵ within time t making q_{h_1} queries, q_{h_2} queries and q_{h_3} queries. We show how to construct an adversary $\mathcal{B}$ to solve the BDH problem.

- **Setup:** The BDH challenger gives an adversary $\mathcal{B}$ the BDH parameters $\langle q, \mathbb{G}_1, \mathbb{G}_1, \hat{e} \rangle$ and an instance $\langle P, aP, bP, cP \rangle$ of the BDH problem. The adversary $\mathcal{B}$ picks a random element s in $\mathbb{Z}_q^*$ and computes $S = sP$. Then he gives the adversary $\mathcal{A}$ `params` $= \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, S, H_1, H_2, H_3 \rangle$ as system parameters for BasicTREPC and a public key $Y = bP$. $\mathcal{B}$ simulates random oracles H_1, H_2, H_3 as follows.
  - H_1**-queries:** $\mathcal{A}$ queries the value of a time t_i to the extraction oracle H_1. To respond to these queries $\mathcal{B}$ maintains a list of tuples $\langle t_i, Q_i \rangle$ called the H_1^{list}. The adversary $\mathcal{B}$ picks a random j where $1 \leq j \leq q_{h_1}$ before responding to H_1-queries. If the query t_i is already queried to H_1, then he returns $Q_i = H_1(t_i)$ in H_1^{list}. Otherwise, he picks a random element Q_i in $\mathbb{G}_1$ and adds $\langle t_i, Q_i \rangle$ to H_1^{list}. Note that $\mathcal{B}$ responds $Q_j = cP$ to the query t_j instead of a randomly selected Q_j.
  - H_2**-queries:** This simulation is also the same as that of Theorem 5.
  - H_3**-queries:** $\mathcal{A}$ queries $\langle g_{R_j}, M_j, U_j, V_j, W_j, Z_j \rangle$ to the random oracle H_3. If this query is already queried to H_3, $\mathcal{B}$ returns σ_j in H_3^{list}. Otherwise, $\mathcal{B}$ randomly picks $\sigma_j \neq \sigma^* \in \{0,1\}^{k_2}$ and returns it.
- **Phase 1:** The adversary $\mathcal{A}$ makes queries to the extraction oracle and the encryption oracle. The adversary $\mathcal{B}$ simulates the oracles to respond the queries as follows.
  - **Extraction-queries:** $\mathcal{A}$ queries t_i to get a timestamp TS_i. $\mathcal{B}$ obtains Q_i such that $H_1(t_i) = Q_i$ running the above algorithm for responding to H_1-queries. Then $\mathcal{B}$ responds $TS_i = sQ_i$ to $\mathcal{A}$.

- **Decryption-queries:** $\mathcal{A}$ queries $\langle t_i, C_i\rangle = \langle t_i, (U_i, V_i, W_i, Z_i, \sigma_i)\rangle$ to the decryption oracle. If $\langle g_{R_i}, M_i, U_i, V_i, W_i, Z_i, \sigma_i\rangle$ does not exist in H_3^{list} then return *Reject*. Otherwise, compute $H_2(g_{R_i})$ by simulating the H_2 oracle and check $W_i = M_i \oplus H_2(g_{R_i})$. If $W_i = M_i \oplus H_2(g_{R_i})$, return M_i and *Reject* otherwise.

- **Challenge:** An adversary $\mathcal{A}$ outputs two equal-length messages (M_0, M_1) and a target release time t_i. If $i \neq j$, then $\mathcal{B}$ reports a failure. The attack on the BDH problem is terminated. Otherwise, he picks a random number $v \in \mathbb{Z}_q^*$, a random elements $g_O \in \mathbb{G}_2$ and two random strings $R \in \{0,1\}^n$, $\sigma^* \in \{0,1\}^{k_2}$, and computes $C = \langle U, V, W, Z, \sigma\rangle = \langle aP, vP, g_O, R, \sigma^*\rangle$. An adversary $\mathcal{B}$ returns C as the challenge to $\mathcal{A}$. Note that σ^* is not returned an output of H_3 queries.
- **Phase 2:** The adversary $\mathcal{A}$ makes extraction queries and decryption queries where $\langle t_i, C_i\rangle \neq \langle t, C\rangle$ as in Phase 1.
- **Guess:** An adversary $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. Then $\mathcal{B}$ picks a random element g_i in H_2^{list} and outputs $W/(g_i \cdot \hat{e}(vS, cP))$ as the solution to the given BDH instance.

If the adversary $\mathcal{B}$ does not report a failure during the simulation, the adversary $\mathcal{A}$'s view is identical to its view in the real attack. Let $\mathcal{H}$ be the event that $\mathcal{A}$ queries the correct g_R to the random oracle H_2. Then $\mathcal{B}$ can derive the correct answer of the BDH problem from g_R in following equtions; $\frac{W}{g_R \cdot \hat{e}(vS,cP)} = \frac{g_R \cdot \hat{e}(aP,bQ_i)\hat{e}(vP,sQ_i)}{g_R \cdot \hat{e}(vS,cP)} = \frac{\hat{e}(aP,bcP)\hat{e}(vsP,Q_i)}{\hat{e}(vS,cP)} = \frac{\hat{e}(aP,bcP)\hat{e}(vS,cP)}{\hat{e}(vS,cP)} = \hat{e}(P,P)^{abc}$.

By the way, in the simulation of the decryption oracle there are cases in which a valid ciphertext is rejected since C_i is rejected if $\langle g_{R_i}, M_i, U_i, V_i, W_i, Z_i, \sigma_i\rangle$ is not in H_3^{list}. One is that σ of the target ciphertext is used as a part σ^* of the decryption query. In this case, the probability that the decryption query is valid is $1/2^{k_2}$. The other is that $\mathcal{A}$ guesses a correct output of H_3 without querying it. This probability is also $1/2^{k_2}$. If the above rejections do not happen, $\mathcal{A}$'s view is identical to its view in the real attack. Let H_3 be the event that $\mathcal{A}$ queries a valid ciphertext without querying to H_3 and ϵ' be the advantage in case that $\mathcal{A}$ is simulated fair. Then ϵ' is computed as follows; $\epsilon' = |Pr[b = b'|\neg \mathsf{H}_3] - 1/2| > |Pr[b = b'] - Pr[\mathsf{H}_3] - 1/2| > (\epsilon - Pr[\mathsf{H}_3])$. Since $\mathcal{A}$ makes at most q_d decryption queries during the simulation, $Pr[\mathsf{H}_3] \leq q_d/2^{k_2}$. Let ϵ'' be the probability that $\mathcal{B}$ outputs the correct answer of the BDH problem when the game fails. We can derive $\epsilon'' = 2\epsilon'/q_{h_2}$ by Lemma 6. In addition, the probability that the adversary $\mathcal{B}$ does not fail during in the simulation is at least $1/g_{h_1}$. Therefore, the advantage of $\mathcal{B}$ that solves the BDH problem is at least $\epsilon''/q_{h_1} = 2\epsilon'/q_{h_1}q_{h_2} = \frac{2}{q_{h_1}q_{h_2}}(\epsilon - \frac{q_d}{2^{k_2}})$. In consequence, we can derive our result $\mathrm{Adv}^{\mathrm{IND-TR-CCA_{OS}}}_{\mathsf{FullTREPC},\mathcal{A}}(k) < \frac{g_{h_1}g_{h_2}}{2} \cdot \mathrm{Adv}^{\mathrm{BDH}}_{\mathcal{G},\mathcal{B}}(k) + \frac{q_d}{2^{k_2}}$ □

In the point of view of an *inside* adversary, BasicTREPC and FullTREPC are not different since he has the public key and secret key pair. Therefore, the following theorem is given from theorem 5 without an additional proof.

Theorem 8. FullTREPC *is secure against IND-TR-CCA$_{IS}$ over the BDH assumption. Namely:* $\mathrm{Adv}^{\mathrm{IND-TR-CCA_{IS}}}_{\mathsf{FullTREPC},\mathcal{A}}(k) \leq \frac{q_{h_1} q_{h_2}}{2} \cdot \mathrm{Adv}^{\mathrm{BDH}}_{\mathcal{G},\mathcal{B}}(k)$.

5 Discussions on TRE-PC

5.1 Reuse of Secret Value for the Release Key

Assume that the sender is a distributer or a company with many users in our schemes. If he selects the secret value v for the release key whenever transmitting a message to users, then large secure storage is required. In our model, the value v for the release key can be reused. First, we consider the case that multi-users have the same release time in applications like distribution system. The sender makes the ciphertext $C = \langle V, C_1, C_2, \ldots, C_n \rangle$ for n users where $C_i = \langle U_i, W_i, Z_i, \sigma_i \rangle$ and broadcasts it. This application is secure since an *inside* adversary cannot break the system by Theorem 5. According to the situation of applications, the sender opens V in a public site as a web page and then he can transmits the ciphertext C_i when the user requests material. The sender just publishes the release key in his site instead of sending the release key to each user for pre-open.

Next, we suppose that the sender sends some ciphertexts $C_1, \ldots, C_m$ with different release times $t_1 \ldots, t_m$ to the receiver where $C_i = \langle V_i, U_i, W_i, Z_i, \sigma_i \rangle$. Even if all V_i are the same; namely the same secret value v for the release key is used, our scheme is secure because the secret value v of the sender and V play a similar role with the master key and the public key of the time server respectively and the release key of a message with a release time t_i can be only used for pre-open of the ciphertext with the release time t_i as a *timestamp*. The release keys for pre-open are respectively different if the release time is different. However, the same secret value v cannot be used for different materials with the same release time since they have the same release key.

5.2 Authenticated TRE-PC

In many applications, to use the TRE scheme the authentication of the sender may be needed for the validity of the ciphertext and the confidence of the release time. We can construct a secure and efficient authenticated TRE-PC (called AuthTREPC) using the efficient signcryption with the bilinear map introduced by Libert and Quisquater [11]. Let the public key and secret key pair of the sender be (x_S, Y_S) and that of the receiver (x_R, Y_R). Then the AuthTREPC scheme is as follows.

- Signcryption: The sender decides a release time t and selects a value $v \in \mathbb{Z}_q$ to make a release key. In addition, he chooses a random value $r \in \mathbb{Z}_q^*$ and a random element $g_R \in \mathbb{G}_2$ and signcrypts a message M as follows; $C = \langle U, V, W, Z \rangle = \langle rP, vP, L \oplus H_2(U, V, Y_R, rY_R), (\sigma || Y_S) \oplus H_3(L) \rangle$ where $g_t = \hat{e}(S, vQ_t)$, $\sigma = M \oplus H(g_t)$, $L = x_S H_1(\sigma, U, V, Y_R)$. The sender sends the C to the receiver over insecure channel.

- Designcryption: When the receiver obtains C, he checks the validity of the ciphertext; $\hat{e}(Y_S, H_1(\sigma', U, V, Y_R)) \stackrel{?}{=} \hat{e}(P, L')$ where $L = W \oplus H_2(U, V, Y_R, x_R U)$ and $(\sigma || Y_S) = Z \oplus H_3(L)$. At time t the receiver obtain TS_t from the time server. Then he can derive a message M from the ciphertext $C = (U, V, W, Z)$ by $M = \sigma \oplus H(g_t)$ where $g_t = \hat{e}(V, TS_t)$. If the sender publishes the release key V_t before the release time, the receiver can obtain M by computing $g_t = \hat{e}(V_t, S)$.

Though this construction requires two pairing operations in signcryption phase, it can check the authenticity of sender when the receiver receives the ciphertext. If we replace g_t and $H_2(V, Y_R, rY_R)$ by $g'_t = \hat{e}(vS + rY_R, Q_i)$ and $H_2(V, Y_R, g'_t)$ respectively in signcryption phase, then the signcryption is performed by one pairing operation. However, in this case the receiver cannot authenticate the ciphertext until the release time passes or the release key is published. While in this case the receiver cannot check if the received message is garbage until the release time passes or the release key is published, AuthTREPC can immediately check the validity of the ciphertext as soon as it is received and requires only one pairing operation to decrypt it after the release time passes or when the release key is received.

6 Application to Certified E-mail System

A fair exchange protocol ensures that either two entities have the expected items or no entity can obtain any information about the other's item after the protocol is complete. In practical environments, to implement the fair exchange, a protocol requires a third party as a trusted arbitrator (TA). There are *on-line* protocols and *off-line* protocols. An on-line protocol is generally difficult to provide the confidentiality since the TA is involved in every transaction. While in an on-line protocol TA plays a role of delivery for processing the protocol, in an off-line protocol TA attends the protocol to solve the dispute only in exceptional circumstances. A certified e-mail system is a practical system providing a fair exchange in which the recipient gets the mail content if and only if the mail provider has the irrefutable receipt on the mail. To construct the secure and efficient certified e-mail system, various protocols have been investigated [1,2,3,13,16]. To be securely used in practical environments, the certified e-mail systems should satisfy *fairness, monotonicity, invisibility of TA, confidentiality*, and *reasonable efficiency* as mentioned in [1,2].

In this section, we introduce how to construct a certified e-mail system based on TRE-PC. A certified e-mail system constructed by TRE-PC is a *communication-efficient off-line system* satisfying the above properties. Considerable off-line certified systems, where TA is involved only in case of the dispute, are introduced in [1,3,13]. We will compare our system based on TRE-PC with them.

Alice		*Bob* (pk_B, sk_B)
$C = \mathsf{Enc}(M, I, v, pk_B)$		
$\sigma_{A1} = \mathsf{Sign}_A(I, C)$	$\xrightarrow{C, \sigma_{A1}}$	
		$\sigma_{A1} \stackrel{?}{=} \mathtt{valid}$
	$\xleftarrow{\sigma_B}$	$\sigma_B = \mathsf{Sign}_B(I, C, \sigma_{A1})$
$\sigma_B \stackrel{?}{=} \mathtt{valid}$		
$rk_I = \mathsf{Gen}_{\mathsf{RK}}(v, I)$		
$\sigma_{A2} = \mathsf{Sign}_A(rk_I)$	$\xrightarrow{rk_I, \sigma_{A2}}$	$M = \mathsf{Dec}(C, rk_I, sk_B)$

Table 1. Certified e-mail system based on TRE-PC in case of fairness

6.1 Certified E-mail System Based on TRE-PC

Certified e-mail system consists of three entities, the mail provider *Alice*, the recipient *Bob*, and the trusted arbitrator *TA*. Certified e-mail system based on TRE-PC is shown in Table 1. First, *Alice* encrypts a mail content by TRE-PC and sends it to *Bob*. *Bob* generates a signature on the received message and gives it to *Alice*. In our system, this signature becomes a receipt on the mail content. After checking a validity of *Bob*' signature, *Alice* gives *Bob* a release key and stores the signature as a receipt if the signature is valid. In case that *Bob* does not receive the release key from *Alice*, he requests arbitration to the *TA* with interchanged messages (see Table 2). *TA* adjudicates on the dispute, and sends a token *td* for decryption (a timestamp in TRE-PC) to *Bob* and *Bob*' signature to Alice. Then *Bob* can obtain a message from the token received from *TA*. While the time server in TRE-PC periodically issues *timestamp*, the *TA* generates the token for decryption only when a player requests it. We define a *token extraction algorithm* as follows.

- $\mathsf{Ext}_{\mathrm{TD}}$: the *token (for decryption) extraction algorithm* used by *TA* takes as input `params`, *TA*' secret key sk_{TA}, identities of two players and state information (A, B, SI), and outputs a token $td_{A,B,SI}$ for decryption where A, B are identities of *Alice* and *Bob*.

Actually, the *token extraction algorithm* is identical with the *timestamp extraction algorithm* except for inputting (A, B, SI) instead of the time t. State information SI should include information on time or a session number, and be different per every transaction. We denote (A, B, SI) as I.

If the protocol is successfully completed, *Alice* and *Bob* exchange a message M and a receipt σ_B on it respectively. If *Alice* does not send a release key rk_I to *Bob* after receiving the receipt σ_B, *Bob* requests arbitration to *TA* and they run the following protocol. *TA* should give a receipt to *Alice* so as to prevent *Bob*'s attempt to successfully retrieve a message without sending a receipt to *Alice*.

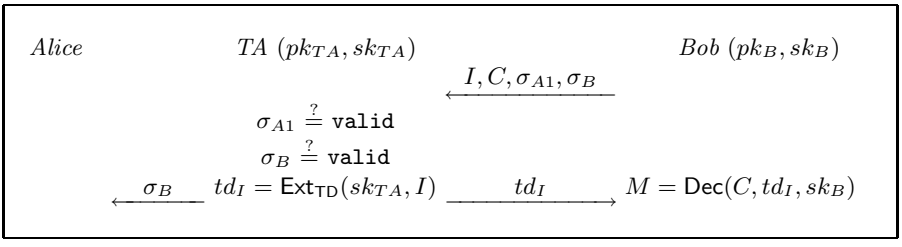

Table 2. Certified e-mail system based on TRE-PC in case of the dispute

6.2 Comparisons

We compare our system based on TRE-PC to previously proposed off-line systems satisfying properties mentioned in Section 6. Asokan *et al.* introduced a secure and efficient fair exchange protocol of digital signatures and applied their protocol to an off-line certified e-mail system in [3]. However, a certified e-mail system based on their protocol is expensive in terms of communication complexity since it uses the cut-and-choose interactive proof technique. Ateniese [1] proposed a certified e-mail system based on verifiable encryption of digital signatures. In his system, the recipient and the TA can be set to be stateless and the recipient can assume a passive role without being involved in the dispute. Micali [13] proposed certified e-mail systems with simple structure. His system is very optimistic in case that the system does not require the confidentiality. Our system based on TRE-PC is very efficient as compared with above systems in regard to communication complexity. Table 3 shows the communication complexity of off-line certified e-mail systems with the confidentiality. In [13], to preserve the confidentiality (or the privacy) the mail provider sends the recipient two encrypted messages. One is a double encrypted message by the public key of the *TA* and that of the recipient and the other is an encrypted message by that of the recipient. Because the length of a mail content is generally much longer than that of others such as |Sign|, $|rk|$ or |VEnc(Sign)|, the system of [13] is inefficient with respect to communication complexity. In addition, when

	Passes	Exchanged data size
[1]	4	$\|\mathrm{Enc}(M)\| + \|\mathrm{VEnc}(\mathrm{Sign}_{\mathrm{RSA}})\| + 3\|\mathrm{Sign}\|$
[13]	3	$2\|\mathrm{Enc}(M)\| + \|\mathrm{Sign}\|$
Our system	3	$\|\mathrm{Enc}(M)\| + 3\|\mathrm{Sign}\| + \|rk\|$

Table 3. Comparisons of communication cost with other certified e-mail systems. (We denote $|x|$ as a bit length of an arbitrary string x and VEnc as a verifiable encryption. Then $|\mathrm{Enc}(M)|$ is the length of a ciphertext, |Sign| is that of a signature, and |VEnc(Sign)| is that of verifiable encryption of a signature.)

1024-bit RSA is used in [1], $|\mathrm{VEnc}(\mathrm{Sign}_{\mathrm{RSA}})| \approx 3000$ bits and $|\mathrm{Sign}| = 1024$ bits because a receipt is a form of RSA signature. However, our system can use the short signature proposed by Boneh *et al.*[7] without additional domain. Their signature is generated in $\mathbb{G}_1$ and $\mathbb{G}_2$ and uses a hash function H_1 defined in TRE-PC. A short signature whose length approximately is 170 bits provides a similar security level to 1024-bit RSA signature. Therefore, in our system $3|\mathrm{Sign}| + |rk|$ is less than 1000 bits when using the short signature.

References

1. G. Ateniese, "Verifiable encryption of digital signatures and applications," *ACM Transactions on Information and System Security*, Vol.7, No.1, pp.1-20, 2004. (Parts of this paper appeared in *ACM CCS 1999* and in *CT-RSA 2002*.)
2. G. Ateniese, B. Medeiros, and M. T. Goodrich, "TRICERT: a distributed certified e-mail scheme," *ISOC NDSS 2001*, 2001.
3. N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," *IEEE Journal on Selected Areas of Communications*, Vol.18, No.4, pp.591-610, 2000. (Extended abstract of this paper appeared in *EUROCRYPT 1998*.)
4. I. F. Blake and A. C-F. Chan, "Scalable, server-passive, user-anonymous timed release public key encryption from bilinear pairing," *Cryptology ePrint Archive*, 2004. Available at `http://eprint.iacr.org/2004/211/`.
5. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schems," *CRYPTO 1998*, LNCS 1462, pp.26-45, 1998.
6. D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," *CRYPTO 2001*, LNCS 2139, pp.213-229, 2001.
7. D. Boneh, B. Lynn, and H. Shacham, "Short signature from the Weil pairing," *ASIACRYPT 2001*, LNCS 2248, pp.514-532, 2001.
8. M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," *ACM CCS 1993*, pp.62-73, 1993.
9. G. D. Crescenzo, R. Ostrovsky, and S. Rajagopalan, "Conditional oblivious transfer and timed-release encryption," *EUROCRYPT 1999*, LNCS 1592, pp.74-89, 1999.
10. A. Joux, "The Weil and Tate pairing as building blocks for public key cryptosystems," *ANTS-V*, LNCS 2369, pp.20-32, 2002.
11. B. Libert and J.-J. Quisquater, "Efficient signcryption with key privacy from Gap Diffie-Hellman groups," *PKC 2002*, LNCS 2947, pp.187-200, 2004.
12. T. C. May, "Timed-releas crypto," *Manuscript*, 1993.
Available at `http://www.cyphernet.org/cyphernomicon/chapter14/14.5.html`.
13. S. Micali, "Simple and fast optimistic protocols for fair electronic exchange," *PODC 2003*, pp.12-19, 2003.
14. I. Osipkov, Y. Kim, and J. H. Cheon, "Timed-release public key based authetnicated encryption," *Cryptology ePrint Archive*, 2004. Available at `http://eprint.iacr.org/2004/231/`.
15. T. Okamoto and D. Pointcheval, "REACT: rapid enhanced-security asymmetric cryptosystem transform," *CT-RSA 2001*, LNCS 2020, pp.159-174, 2001.
16. B. Pfitzmann, M. Schunter, and M. Waidnet, "Optinal efficiency of optimistic contract signing," *PODC 1998*, pp.113-122, 1998.
17. R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto.," *MIT LCS Tech. Report MIT/LCS/TR-684*, 1996.