Revisiting the Security Model for Timed-Release Public-Key Encryption with Pre-Open Capability

Alexander W. Dent and Qiang Tang
Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
{a.dent, qiang.tang}@rhul.ac.uk

June 28, 2007

Abstract

In this paper we investigate a security model for Timed-Release Encryption schemes with Pre-Open Capability (TRE-PC schemes) proposed by Hwang, Yum, and Lee. Firstly, we show that the HYL model possesses a number of defects and fails to model some potentially practical security vulnerabilities faced by TRE-PC schemes. Secondly, we propose a new security model for TRE-PC schemes which models the securities against four kinds of attacker and avoids the defects of the HYL model. We also work out the complete relations among the security notions defined in the new model. Thirdly, we introduce the notion of TRE-PC KEM, which is a special type of KEM, and show a way to construct a TRE-PC scheme using a TRE-PC KEM and a DEM. Finally, we propose an instantiation of TRE-PC KEM and prove its security.

Contents

1 Introduction

The concept of Timed-Release Encryption (TRE), i.e. sending a message which can only be decrypted after a certain release time, is attributed to May [15]. Later on, Rivest, Shamir, and Wagner further elaborated on this concept and gave a number of its applications including electronic auctions, key escrow, chess moves, release of documents over time, payment schedules, press releases and etc. [17]. It is worth mentioning that some other timed primitives have been developed, for example, "price via processing" by Dwork and Naor [13], timed key escrow by Bellare and Goldwasser [1, 2], and timed commitments by Boneh and Noar [5].

Informally, a TRE scheme is a public-key encryption scheme which achieves the following two goals:

1. An outside attacker cannot decrypt the ciphertext. An outside attacker is any third-party entity besides the sender and receiver.

2. A legitimate receiver can only decrypt the ciphertext after the release time.

In the literature, there are two approaches used to construct TRE schemes. One approach is based on the time-lock puzzle technique which was originally proposed by Merkle [16] to protect communications against passive adversaries. This technique was then extended in [7, 17] to construct TRE schemes. The idea of this approach is that a secret is transformed in such a way that all kinds of machines (serial or parallel) take at least a certain amount of time to solve the underlying computational problems (puzzle) in order to recover the secret. The release time is equal to the time at which the puzzle is released plus the minimum amount of time that it would take to solve the puzzle. However, this means that not all users are capable of decrypting the ciphertext at the release time as they may have different computational power. The other approach is to use a trusted time server, which, at an appointed time, will assist in releasing a secret to help decrypt the ciphertext (e.g. [6, 17]). Generally, time-server-based schemes require interaction between the server and the users, and should prevent possible malicious behaviour of the time server.

In standard TRE schemes, the receiver can only decrypt the ciphertext at (or after) the release time. If the sender changes its mind after the ciphertext is sent, and wishes the receiver to decrypt the message immediately, then the only thing that the sender can do is to re-send the plaintext to the receiver in such a way that the receiver can immediately decrypt the message. However, in some circumstances, we may need a special kind of TRE schemes, in which a mechanism enables the receiver to decrypt the ciphertext before the release time without requiring the sender to re-send the plaintext. Recently, Hwang, Yum, and Lee [14] extended the concept of TRE schemes and proposed a security model (referred to as the HYL model) for TRE schemes with Pre-Open Capability (TRE-PC). Informally, a TRE-PC scheme is a public-key encryption scheme which achieves the following goals:

1. At any time, an outside attacker cannot decrypt the ciphertext,

2. At (or after) the release time, when the plaintext is intended to be released, the receiver can decrypt the ciphertext. However, before the release time, the receiver cannot decrypt the ciphertext.

3. Before the release time, the sender can help the receiver to decrypt the ciphertext by publishing a pre-open key.

In the HYL model, a trusted time server is required to periodically issue a timestamp, but real-time interaction between the trusted time server and the users is not needed.

As a special type of TRE scheme, a TRE-PC scheme is always a possible substitute of a standard TRE scheme in all the possible applications where the latter is used. In fact, we can argue that TRE-PC scheme is more suitable than the general TRE scheme in many applications where the Pre-Open capacity is potentially needed. Taking the electronic auction as an example, normally a group of bidders in an auction seal their bid so that it can be opened after the bidding period is closed. However, if all of these bidders want to open the bid at some point before the pre-defined open time, then they may come across some problems if a standard TRE scheme is adopted. For instance, if the TRE scheme proposed in [6] is used, the trusted time server cannot simply release the time-specific trapdoor because the information may also be used by other bids. However, if a TRE-PC scheme is adopted, then the bidders just need to release their pre-open key to get the job done. Moreover, if the TRE-PC scheme is binding (defined in Section 3.2) then it is guaranteed that no bidders can maliciously open a different value which is different from that it originally sent.

The first contribution of this paper is that we analyse the HYL model and show that it fails to model some practical attacks faced by TRE-PC schemes. Specifically, we show that the HYL model has not modeled the attacks imposed by an outside attacker which knows the time server's secret key, although it claimed to achieve this. We further show that the HYL model has not modelled the attacks imposed by a malicious sender which may abuse the protocol executions.

The second contribution of this paper is that we propose a new security model for TRE-PC schemes, which models four kinds of potential adversaries: the malicious outside attacker which does not know the time server's secret key, the curious time server (which is one kind of outside attacker), the malicious sender, and the curious receiver. The proposed model avoids the defects possessed by the HYL model and provides a valuable framework to evaluate the security of TRE-PC schemes. Moreover, we work out the complete relations among the security notions defined in the new model.

The third contribution of this paper is that we introduce the notion of TRE-PC KEM, which can be thought of as combining the functionality of a KEM and a TRE-PC scheme, show a way to construct a TRE-PC scheme using TRE-PC KEM and a DEM, and provide the security criteria for the TRE-PC KEM and

the DEM. In addition, we also propose an instantiation of a TRE-PC KEM and prove its security.

In fact, our work about the TRE-PC construction follows the basic idea of the KEM-DEM model for hybrid encryption algorithms [8, 18], which splits a hybrid encryption scheme into two distinct components: an asymmetric Key Encapsulation Mechanism (KEM) and a symmetric Data Encapsulation Mechanisms (DEM). The KEM-DEM has the advantage that it allows the security requirements of the asymmetric and symmetric parts of the scheme to be completely separated and studied independently. In addition, it also enables us to construct public-key encryption schemes by instantiating KEM and DEM separately. Due to this, this paradigm has been extended to the signcryption setting by proposing new security criteria for the KEM and the DEM. Some examples are the work by Dent [11, 12] and the work by Bjørstad and Dent [4]. In [10] Dent propose a number of generic constructions for provably secure KEMs. In [3] Bentahar et al. extend the concept of key encapsulation mechanisms to the primitives of ID-based and certificateless encryption. A TRE-PC KEM is indeed a special type of KEM, which is specially designed for the TRE-PC setting. By exploiting the KEM-DEM model, our construction inherently possesses similar advantages. However, the KEM-DEM model does not model all possible hybrid encryption schemes, which implies that there may exist elegant TRE-PC constructions other than ours.

The rest of this paper is organised as follows. In Section 2 we analyse the HYL model and schemes. In Section 3 we propose a new security model for TRE-PC schemes. In Section 4 we investigate the relations among the security notions developed in Section 3. In Section 5 we describe the notion of TRE-PC KEM and define its security. In Section 6 we show a way to construct TRE-PC schemes using TRE-PC KEM and DEM, and provide a security analysis for our construction. In Section 7 we propose an instantiation of TRE-PC KEM and provide its security proofs. In Section 8 we conclude this paper.

2 Review of the HYL model and schemes

2.1 Description of the HYL model

In the HYL model [14], two kinds of entities are involved in a TRE-PC scheme: a trusted time server and users. The trusted time server publishes timestamps periodically, and every user may act as either a sender or a receiver. The following two kinds of attacker are claimed to be considered:

- An outside attacker without the receiver's private key, which models either a dishonest time server or an eavesdropper who tries to decrypt the legal receiver's ciphertext.
- An inside attacker with the receiver's private key, which models a legal receiver who tries to decrypt the ciphertext before the release time without the pre-open key.

In the HYL model, a TRE-PC scheme consists of the following 6 polynomial-time algorithms:

- Setup: The setup algorithm takes a security parameter 1^ℓ as input, and returns the master key mk and the system parameters $param$, where mk is kept secret by the time server and $param$ is published.
- $\mathsf{Ext_{TS}}$: Run by the trusted time server, this timestamp extraction algorithm takes the system parameters $param$, the master key mk, and the release time t as input, and returns the timestamp TS_t. The time server is required to publish TS_t at time t.
- $\mathsf{Gen_{PK}}$: Run by a user, this key generation algorithm takes a security parameter 1^ℓ and the system parameters $param$ as input, and returns a public/private key pair (pk_r, sk_r).
- Enc: Run by a sender, this encryption algorithm takes a message m, a release time t, and a randomly-chosen pre-open secret value v, and returns a ciphertext C.
- $\mathsf{Gen_{RK}}$: Run by a sender, this pre-open key generation algorithm takes the pre-open secret value v and the release time t as input, and returns the pre-open key V_t.
- Dec: Run by a receiver, this decryption algorithm runs in two modes. Before the release time, on input of the pre-open key V_t, the ciphertext C, and the receiver's private key, this algorithm returns either the plaintext or an error message. Otherwise, on the input of the timestamp TS_t, the ciphertext C, and the receiver's private key, this algorithm returns either the plaintext or an error message.

In the HYL model, three security properties are modelled, although we show later that some other security properties (see in Section 3) also need to be modelled. The modelled securities are: the security under a chosen ciphertext attack against outside adversaries (IND-TR-CCA$_{\mathrm{OS}}$ security), the security under a chosen plaintext attack against outside adversaries (IND-TR-CPA$_{\mathrm{OS}}$ security), and the security under a chosen ciphertext attack against inside adversaries (IND-TR-CCA$_{\mathrm{IS}}$ security).

Definition 1. *Suppose $\mathcal{A}$ is a polynomial-time outside attacker, then a TRE-PC scheme E is IND-TR-CCA$_{OS}$ secure if $\mathcal{A}$ only has negligible advantage in the following game.*

1. Game setup: The challenger takes a security parameter 1^ℓ and runs Setup to generate the master key mk and the system parameters $param$. The challenger also runs $\mathsf{Gen_{PK}}$ to generate a public/private key pair (pk_r, sk_r).
2. Phase 1: The attacker $\mathcal{A}$ takes $(pk_r, param)$ as input, and can make the following queries.

- $\mathcal{A}$ makes timestamp extraction queries on any time t. On receiving a query, the challenger runs the $\mathsf{Ext_{TS}}$ and returns the output.
- $\mathcal{A}$ makes decryption queries on (t, C). On receiving a query, the challenger runs the Dec and returns the output.

3. Challenge: $\mathcal{A}$ selects two equal length messages m_0, m_1 and a release time t^*. The challenger picks a random bit b, encrypts m_b for release at time t^*, and returns the ciphertext C^*.

4. Phase 2: $\mathcal{A}$ can continue to make extraction and decryption queries as in Phase 1. However, $\mathcal{A}$ is not allowed to make a decryption query on (t^*, C^*).

5. Guess: $\mathcal{A}$ outputs a guess bit b'.

In this game the attacker wins the game if $b = b'$. The attacker's advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$.

Definition 2. *A TRE-PC scheme E is said to be IND-TR-CPA$_{OS}$ secure if it is IND-TR-CCA$_{OS}$ secure against adversaries that make no decryption queries.*

In [14] the attack game for IND-TR-CCA$_{\mathrm{IS}}$ security is informally defined; but we reconstruct a formal definition as follows.

Definition 3. *Suppose $\mathcal{A}$ is a polynomial-time inside attacker, then a TRE-PC scheme E is IND-TR-CCA$_{IS}$ secure if $\mathcal{A}$ only has negligible advantage in the following game.*

1. Game setup: The challenger takes a security parameter 1^ℓ and runs Setup to generate the master key mk and the system parameters $param$. The challenger runs $\mathsf{Gen_{PK}}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker $\mathcal{A}$ takes $(pk_r, sk_r, param)$ as input, and can make timestamp extraction queries on any time t. On receiving a query, the challenger runs the $\mathsf{Ext_{TS}}$ and returns the output.

3. Challenge: $\mathcal{A}$ selects two equal length messages m_0, m_1 and a release time t^*, which has not been queried to the $\mathsf{Ext_{TS}}$ oracle. The challenger picks a random bit b, encrypts m_b for release at time t^*, and returns the ciphertext C^*.

4. Phase 2: $\mathcal{A}$ continues to make the same kinds of queries as in Phase 1. However, $\mathcal{A}$ is not allowed to make an $\mathsf{Ext_{TS}}$ query on the time t^*.

5. Guess: $\mathcal{A}$ terminates by outputting a guess bit b'.

In this game the attacker wins the game if $b = b'$. The attacker's advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$.

2.2 Remarks on the HYL model

We have the following remarks on the HYL model.

1. In the HYL model, the decryption process is described by one single algorithm, which, however, works in two different modes depending on the input. Therefore, it is appropriate to formalise the decryption process with two independent algorithms.

2. In a security model for TRE-PC schemes, it should be assumed that any outside attacker will have access to the pre-open key for the challenge ciphertext. This models the real-world situation where the outside attacker observes the release key as it is being sent to the receiver after the sender chooses to allow the receiver to pre-open the ciphertext. However, in the HYL model, it is not explicitly specified that the attacker has access to the pre-open key.

3. In the HYL model, $\mathsf{Gen_{RK}}$ is formalised but never used in the security model. One possible way to eliminate this defect is allowing the attacker to access the $\mathsf{Gen_{RK}}$ oracle (as a result the above defect can also be eliminated). Alternatively, we can absorb the functionality of $\mathsf{Gen_{RK}}$ into Enc by requiring the latter to output both the ciphertext and the pre-open key (as depicted in the new model).

4. In the HYL model, the Enc algorithm does not take the receiver's public key as input, however, it is obvious that the receiver's public key should be part of the input. We accept that this is probably just a typographical error.

5. In the attack game for the IND-TR-CCA$_{\mathrm{IS}}$ security, the attacker $\mathcal{A}$ is allowed to make $\mathsf{Ext_{TS}}$ queries on any release time t except t^*. However, in reality a receiver only needs to mount this attack before the release time t^* since it will decrypt the ciphertext with the timestamp TS_{t^*} at (and after) t^*. Therefore, the attacker $\mathcal{A}$ should only be allowed to make $\mathsf{Ext_{TS}}$ queries on any time t which is smaller than the release time t^*.

6. In the HYL model the authors claimed that the outside attacker models either a dishonest time server or an eavesdropper who tries to decrypt the legal receiver's ciphertext. However, a malicious time server has not been considered in the attack game which is used to evaluate IND-TR-CCA$_{\mathrm{OS}}$ security, because the attacker has no knowledge of the master key mk. As a result, a TRE-PC scheme, which is proved IND-TR-CCA$_{\mathrm{OS}}$ secure, might be insecure against a malicious attacker which knows the time server's secret key.

7. As one of the objectives of a TRE-PC scheme, the sender can enable the receiver to decrypt a ciphertext before its release time. In some circumstances, the sender may wish to make the receiver decrypt a false message different from which was originally sent, by sending a false pre-open key

to the receiver. An example attack is shown in Section 2.3. To make the TRE-PC scheme work properly, the malicious sender should also be considered in the security model. In the new security model, we define a TRE-PC scheme to be binding if it is secure against this malicious sender attack.

2.3 Analysis of the HYL basic scheme

2.3.1 Description of the scheme

In [14] Hwang, Yum, and Lee proposed the following TRE-PC scheme (HYL basic scheme) which is claimed to be IND-TR-CPA$_{\mathrm{OS}}$ and IND-TR-CCA$_{\mathrm{IS}}$ secure. The scheme works as follows.

- Setup: Given a security parameter 1^ℓ, the following parameters are generated:
  - two multiplicative groups $\mathbb{G}_1$, $\mathbb{G}_2$ of prime order q, and a generator P of $\mathbb{G}_1$,
  - a polynomial-time computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$,
  - two hash functions $h_1 : \{0,1\}^* \to \mathbb{G}_1$, $h_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some n,
  - a master key $s \in \mathbb{Z}_q$ for the time server, and the public key $S = sP$.

  The message space and the ciphertext space are $\{0,1\}^n$ and $\mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^n$, respectively. The parameters $param = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, S, n, h_1, h_2)$ are published.

- $\mathsf{Ext_{TS}}$: At the time t, the time server computes $Q_t = h_1(t)$ and publishes the timestamp $TS_t = sQ_t$.

- $\mathsf{Gen_{PK}}$: A user runs this algorithm to generate its public/private key pair (Y, x), where $x \in_R \mathbb{Z}_q$ and $Y = xP$.

- Enc: Suppose a sender wishes to send a message m, it first selects a release time t and a secret value $v \in_R \mathbb{Z}_q$, and outputs the ciphertext $C = (rP, vP, m \oplus h_2(g_t))$ where $g_t = \hat{e}(rY + vS, Q_t)$ and $r \in_R \mathbb{Z}_q$.

- $\mathsf{Gen_{RK}}$: When the sender wants the ciphertext C to be decrypted before the release time t, it computes and publishes the pre-open key $V_t = vQ_t$.

- Dec: Before the release time, given the pre-open key V_t, the receiver decrypts $C = (R, V, W)$ by computing

$$m = W \oplus h_2(\hat{e}(R, xQ_t)\hat{e}(V_t, S))$$

  Otherwise, given the timestamp TS_t, the receiver decrypts $C = (R, V, W)$ by computing

$$m = W \oplus h_2(\hat{e}(R, xQ_t)\hat{e}(V, TS_t))$$

2.3.2 Analysis results

We show that a malicious sender can mount an attack to make the receiver decrypt a false message, which is not the message the sender originally sent.

Suppose the sender sent the encrypted message $C = (rP, vP, m \oplus h_2(g_t))$, and then before the release time t, it publishes a false pre-open key $V_t' \neq V_t$. With V_t', the receiver will compute the plaintext m' as:

$$\begin{aligned} m' &= m \oplus h_2(g_t) \oplus h_2(\hat{e}(R, xQ_t)\hat{e}(V_t', S)) \\ &= m \oplus h_2(\hat{e}(R, xQ_t)\hat{e}(V_t, S)) \oplus h_2(\hat{e}(R, xQ_t)\hat{e}(V_t', S)) \end{aligned}$$

It is straightforward to verify that the probability the following equation holds is negligible

$$h_2(\hat{e}(R, xQ_t)\hat{e}(V_t, S)) = h_2(\hat{e}(R, xQ_t)\hat{e}(V_t', S)) \,,$$

so that the probability $m = m'$ is also negligible.

It should be noted that the the other HYL scheme (referred to the full TRE-PC scheme), which is claimed to be IND-TR-CCA$_{\mathrm{OS/IS}}$ secure in [14], appears to be resistant to malicious sender attack because the receiver will check the validity of the plaintext at the end of the decryption.

Besides this security vulnerability, the decryption algorithm of this scheme does not satisfy the definition in the HYL model. In the decryption algorithm, t should be included in the inputs because $Q_t = h_1(t)$ is used in the computation, however t is not required as an input for the decryption algorithm in the HYL model. In fact, this inconsistency also occurs in the full TRE-PC scheme.

3 A new security model for TRE-PC schemes

We propose a new security model for the TRE-PC schemes. As in the HYL model [14], the following two kinds of entities are involved in a TRE-PC scheme:

- The users, every one of which may act as both a sender and a receiver.
- A trusted time server, which is required to publish timestamps periodically. We assume that the time server acts properly in generating its parameters and publishing the timestamps. However, when discussing the security, we take into account the fact that the time server may be curious, i.e. it may try to decrypt the ciphertext. Except this, the time server will do nothing else malicious.

In the proposed model, we consider the following four kinds of adversaries:

- Outside adversaries who do not know the master key of the time server. In the rest of this paper, the term *outside attacker* refers to this kind of attacker, while the term *curious time server* (see below) refers to the special kind of outside attacker which knows the master key of the time server.

- The curious time server who knows the master key of the time server.
- Legal but curious receivers who try to decrypt the ciphertext before the release time without the pre-open key.
- Legal but malicious senders who try to make the receiver recover a false message different from which was originally sent.

Generally, a TRE-PC scheme consists of the following 6 polynomial-time algorithms.

1. Setup: Run by the time server, this setup algorithm takes a security parameter 1^ℓ as input, and generates a secret master-key mk and the global parameters $param$. We assume that all subsequent algorithms takes $param$ implicitly as an input.
2. $\mathsf{Gen_U}$: Run by a user, this user key generation algorithm takes a security parameter 1^ℓ as input, and generates a public/private key pair (pk_r, sk_r).
3. $\mathsf{Ext_{TS}}$: Run by the time server, this timestamp extraction algorithm takes mk and a time t as input, and generates a timestamp TS_t for the time t.
4. Enc: Run by a sender, this encryption algorithm takes a message m, a release time t, and the receiver's public key as input, and returns a ciphertext C and its pre-open key V_C. It should be noted that initially the sender should send the ciphertext C in company with the release time t to the receiver, therefore the receiver can know the release time of C. The sender stores the the pre-open key V_C and publishes it when pre-opening the ciphertext C.
5. $\mathsf{Dec_{RK}}$: Run by the receiver, this decryption algorithm takes a ciphertext C, the pre-open key V_C, and the receiver's private key as input, and returns either the plaintext or an error message ($\perp$). In reality, the receiver can only run this algorithm after the sender releases the pre-open key V_C.
6. $\mathsf{Dec_{PK}}$: Run by the receiver, this decryption algorithm takes a ciphertext C, a timestamp TS_t which is determined by the release time accompanied with C, and the receiver's private key as input, and returns either the plaintext or an error message ($\perp$).

3.1 Soundness of a TRE-PC scheme

Informally, the decryption algorithms of a sound TRE-PC scheme should always "undo" the output of the encryption algorithm. Formally, soundness is defined as follows.

Definition 4. *A TRE-PC scheme E is sound if, for any time t, message m, and (K, C) where*

$$(C, V_C) = \mathsf{Enc}(m, t, pk_r),$$

the following two requirements are satisfied

$$m = \mathsf{Dec}_{\mathsf{RK}}(C, V_C, sk_r),$$

$$m = \mathsf{Dec}_{\mathsf{PK}}(C, TS_t, sk_r).$$

3.2 Binding of a TRE-PC scheme

As we have pointed out in the previous analysis, a sender may act maliciously when it tries to pre-open the ciphertext. In order to make a TRE-PC scheme work correctly, this type of malicious sender attack should be prevented. We give the following definition of *binding* to formalise this property.

Definition 5. *A TRE-PC scheme E is binding if any polynomial-time attacker $\mathcal{A}$ only has a negligible probability of winning the following game.*

1. Game setup: The challenger runs Setup to generate the time server's master key mk and the public system parameters: $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).

2. Challenge: The attacker $\mathcal{A}$ executes with input $(pk_r, param)$. At some point, $\mathcal{A}$ generates a ciphertext C^* for release at time t^* and a pre-open key V_{C^*}, and then terminates by outputting (C^*, t^*, V_{C^*}). During its execution, $\mathcal{A}$ has access to the following oracles:

   - An oracle for $\mathsf{Ext_{TS}}$, which, on receiving a query for time t, returns $\mathsf{Ext_{TS}}(mk, t)$.
   - An oracle for $\mathsf{Dec_{RK}}$, which, on receiving a query for (C, V'_C), returns $\mathsf{Dec_{RK}}(C, V'_C, sk_r)$. Note that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key for C.
   - An oracle for $\mathsf{Dec_{PK}}$, which, on receiving a query for (C, t'), where t' may not be the release time for C, returns $\mathsf{Dec_{PK}}(C, TS_{t'}, sk_r)$.

Suppose $O_1 = \mathsf{Dec_{RK}}(C^*, V_{C^*}, sk_r)$ and $O_2 = \mathsf{Dec_{PK}}(C^*, TS_{t^*}, sk_r)$. In this game $\mathcal{A}$ wins if $O_1 \neq \perp$, $O_2 \neq \perp$, and $O_1 \neq O_2$.

It is worth stressing that we have adopted the term "binding", which is also used to describe a property of commitment schemes, such as that descrribed in [5]. The binding property for a TRE-PC scheme guarantees that, if the attacker has encrypted some message, then it cannot release a pre-open key to force the receiver to decrypt a message which is different from that which was originally sent. This is clearly analogous to the binding property in commitment schemes. The difference is that explicit proofs are usually required in commitment schemes, while no such proofs are required in a TRE-PC scheme (as shown below). We further point out that, if the receiver obtains $\perp$ as a result of the decryption procedure, then it can confirm that the sender has malfunctioned. The formalisation of ciphertext validity, as that in [9], is outside the scope of this paper.

In fact, the binding property for a TRE-PC scheme also relates to the secure transportation of the pre-open key, when the sender decides to open the encrypted message before the pre-defined release time. If the TRE-PC scheme is binding, then the pre-open key does not need to be integrity protected; otherwise, the pre-open key should be integrity protected to guarantee that the receiver will obtain the message which the sender intended to send.

3.3 Security against malicious outsiders

In this subsection we define the security of a TRE-PC scheme against outside adversaries which do not know the time server's master key. Specifically, we define the security under an adaptive chosen ciphertext attack (IND-TR-CCA$_{\mathrm{OS}}$) and the security under an adaptive chosen plaintext attack (IND-TR-CPA$_{\mathrm{OS}}$).

Definition 6. *A TRE-PC scheme E is IND-TR-CCA$_{OS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ only has negligible advantage in the following game.*

1. Game setup: The challenger runs Setup to generate the time server's master key mk and the public system parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input $(pk_r, param)$. $\mathcal{A}_1$ has access to the following oracles.

   - $\mathsf{Ext_{TS}}$ oracle, which, on receiving a query for time t, returns $\mathsf{Ext_{TS}}(mk, t)$.
   - $\mathsf{Dec_{RK}}$ oracle, which, on receiving a query for (C, V'_C), returns $\mathsf{Dec_{RK}}(C, V'_C, sk_r)$. It should be noted that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key of C.
   - $\mathsf{Dec_{PK}}$ oracle, which, on receiving a query for (C, t'), returns $\mathsf{Dec_{PK}}(C, TS_{t'}, sk_r)$. It should be noted that t' need not be the legitimate release time for C.

   $\mathcal{A}_1$ terminates by selecting two equal length messages m_0, m_1 and a release time t^*, and outputting (m_0, m_1, t^*). In addition, $\mathcal{A}_1$ also outputs some state information $state$.

3. Challenge: The challenger picks a random bit $b \in \{0, 1\}$, and returns $(C^*, V_{C^*}) = \mathsf{Enc}(m_b, t^*, pk_r)$.

4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same kinds of oracles as those in Phase 1. But $\mathcal{A}_2$ is not allowed to make the following two queries: a query to the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*), and a query to the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}).

   $\mathcal{A}_2$ terminates by outputing a guess bit $b' \in \{0, 1\}$.

In this game $\mathcal{A}$ wins the game if $b = b'$, i.e. $\mathcal{A}$'s advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$.

Definition 7. *A TRE-PC scheme E is IND-TR-CPA$_{OS}$ secure if it is IND-TR-CCA$_{OS}$ secure against adversaries that make no decryption queries.*

3.4 Security against curious time server

In this subsection we define the security of a TRE-PC scheme against the curious time server. Specifically, we define the security under an adaptive chosen ciphertext attack (IND-TR-CCA$_{\mathrm{TS}}$) and the security under an adaptive chosen plaintext attack (IND-TR-CPA$_{\mathrm{TS}}$).

Definition 8. *A TRE-PC scheme E is IND-TR-CCA$_{TS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ only has negligible advantage in the following game.*

1. Game setup: The challenger runs Setup to generate the time server's master key mk and the public system parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input $(mk, pk_r, param)$. $\mathcal{A}_1$ has access to the following oracles.

   - $\mathsf{Dec_{RK}}$ oracle, which, on receiving a query for (C, V'_C), returns $\mathsf{Dec_{RK}}(C, V'_C, sk_r)$. It should be noted that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key of C.
   - $\mathsf{Dec_{PK}}$ oracle, which, on receiving a query for (C, t'), returns $\mathsf{Dec_{PK}}(C, TS_{t'}, sk_r)$. It should be noted that t' need not be the legitimate release time for C.

   $\mathcal{A}_1$ terminates by selecting two equal length messages m_0, m_1 and a release time t^*, and outputting (m_0, m_1, t^*). In addition, $\mathcal{A}_1$ also outputs some state information $state$.

3. Challenge: The challenger picks a random bit $b \in \{0, 1\}$, and returns $(C^*, V_{C^*}) = \mathsf{Enc}(m_b, t^*, pk_r)$.

4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same kinds of oracles as those in Phase 1. But $\mathcal{A}_2$ is not allowed to make the following two queries: a query to the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*), and a query to the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}).

   $\mathcal{A}_2$ terminates by outputing a guess bit $b' \in \{0, 1\}$.

In this game $\mathcal{A}$ wins the game if $b = b'$, i.e. $\mathcal{A}$'s advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$.

Definition 9. *A TRE-PC scheme E is IND-TR-CPA$_{TS}$ secure if it is IND-TR-CCA$_{TS}$ secure against adversaries that make no decryption queries.*

3.5 Security against malicious receiver

If a TRE-PC scheme is to be deemed secure, then it should resist attacks in which the legitimate receiver attempts to decrypt the ciphertext before the release time and without the pre-open key. In this case, since the attacker will know the receiver's private key, it does not make sense to distinguish between CCA and CPA notions of security. However, for the sake of consistency, we will use the notion "IND-TR-CPA$_{\mathrm{IS}}$" to describe the security of the scheme against a malicious receiver (an inside adversary).

Definition 10. *A TRE-PC scheme E is IND-TR-CPA$_{IS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ only has negligible advantage in the following game.*

1. Game setup: The challenger runs Setup to generate the time server's master key mk and the public system parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$. $\mathcal{A}_1$ has access to the $\mathsf{Ext_{TS}}$ oracle, which, on receiving a query for time t, returns $\mathsf{Ext_{TS}}(mk, t)$. $\mathcal{A}_1$ terminates by outputting two equal length messages m_0, m_1 and a release time t^* which is larger than all the inputs to the $\mathsf{Ext_{TS}}$ oracle. In addition, $\mathcal{A}_1$ also outputs some state information $state$.

3. Challenge: The challenger picks a random bit $b \in \{0, 1\}$, computes $(C^*, V_{C^*}) = \mathsf{Enc}(m_b, t^*, pk_r)$, and returns C^*.

4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input $(C^*, state)$. $\mathcal{A}_2$ has access to the $\mathsf{Ext_{TS}}$ oracle, which, on receiving a query for time $t < t^*$, returns $\mathsf{Ext_{TS}}(mk, t)$.

   $\mathcal{A}_2$ terminates by outputting a guess bit $b' \in \{0, 1\}$.

In this game $\mathcal{A}$ wins the game if $b = b'$, i.e. $\mathcal{A}$'s advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$.

4 Relations among security notions

We first give two relation definitions: "A $\longrightarrow$ B" means that if a scheme is secure in the sense of A then it is secure in the sense of B; "A $\not\longrightarrow$ B" means that, even if a scheme is secure in the sense of A , it may not be secure in the sense of B. In other words, "A $\not\longrightarrow$ B" means that we can construct a scheme which is secure in the sense of A but not secure in the sense of B. It is clear to see that the "$\longrightarrow$" relation is transitive, which means that if A $\longrightarrow$ B and B $\longrightarrow$ C then A $\longrightarrow$ C.

We prove that the relations described in the following figure hold for the security notions, where the arrow represents the relation "$\longrightarrow$" while the hatched

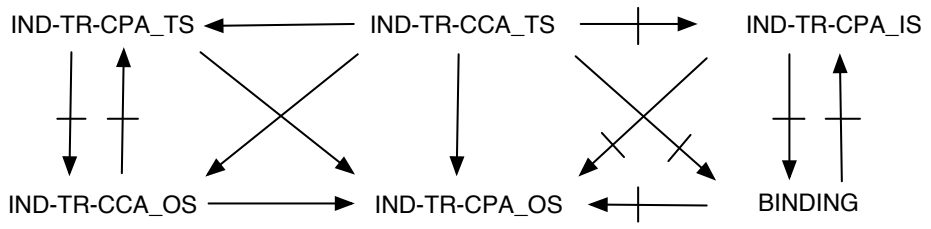

Figure 1: Relations among the security notions

arrow represents the relation "$\not\longrightarrow$". Given these relations, we are able to deduce the relation between any two security notions.

By the definitions in Section 3, it is easy to see that the following relations hold.

1. IND-TR-CCA$_{\mathrm{TS}} \longrightarrow$ IND-TR-CCA$_{\mathrm{OS}} \longrightarrow$ IND-TR-CPA$_{\mathrm{OS}}$
2. IND-TR-CCA$_{\mathrm{TS}} \longrightarrow$ IND-TR-CPA$_{\mathrm{TS}} \longrightarrow$ IND-TR-CPA$_{\mathrm{OS}}$

Hence, in the rest of this section we only need to prove that other relations hold among the security notions. We implicitly assume throughout that a TRE-PC scheme is sound.

4.1 Relations between IND-TR-CPA$_{\mathrm{TS}}$ and IND-TR-CCA$_{\mathrm{OS}}$

We prove that the following relations hold between IND-TR-CPA$_{\mathrm{TS}}$ and IND-TR-CCA$_{\mathrm{OS}}$.

- IND-TR-CPA$_{\mathrm{TS}} \not\longrightarrow$ IND-TR-CCA$_{\mathrm{OS}}$
- IND-TR-CCA$_{\mathrm{OS}} \not\longrightarrow$ IND-TR-CPA$_{\mathrm{TS}}$

Theorem 1. *If there exists an IND-TR-CPA$_{TS}$ secure TRE-PC scheme E, then there exists a TRE-PC scheme E' which is IND-TR-CPA$_{TS}$ secure but not IND-TR-CCA$_{OS}$ secure.*

Proof. Suppose $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ is an IND-TR-CPA$_{\mathrm{TS}}$ secure TRE-PC scheme, then we can construct a new TRE-PC scheme $E' = (\mathsf{Setup}', \mathsf{Gen}'_{\mathsf{U}}, \mathsf{Ext}'_{\mathsf{TS}}, \mathsf{Enc}', \mathsf{Dec}'_{\mathsf{RK}}, \mathsf{Dec}'_{\mathsf{PK}})$, where the algorithms are defined as follows:

- The algorithms Setup',$\mathsf{Gen}'_{\mathsf{U}}$,$\mathsf{Ext}'_{\mathsf{TS}}$ are defined in the same way as in E.
- $\mathsf{Enc}'(m, t, pk'_r) = (C||0, V_C)$, where $(C, V_C) = \mathsf{Enc}(m, t, pk'_r)$.
- $\mathsf{Dec}'_{\mathsf{RK}}(C||b, V_C, sk'_r) = \mathsf{Dec}_{\mathsf{RK}}(C, V_C, sk'_r)$, where $b \in \{0, 1\}$.
- $\mathsf{Dec}'_{\mathsf{PK}}(C||b, TS_t, sk'_r) = \mathsf{Dec}_{\mathsf{PK}}(C, TS_t, sk'_r)$, where $b \in \{0, 1\}$.

The validity of this theorem lies in the following two lemmas.

Lemma 1. *E' is IND-TR-CPA$_{TS}$ secure.*

Proof. Suppose an IND-TR-CPA$_{\mathrm{TS}}$ attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ has the advantage δ in attacking E'. We show that there exists an IND-TR-CPA$_{\mathrm{TS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that also has advantage δ. Hence, we will be able to conclude that δ is negligible as E is IND-TR-CPA$_{\mathrm{TS}}$ secure.

The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public parameters $param$, the public key pk_r, and the master key mk.
2. $\mathcal{A}_1$ sets $param' = param$, $pk'_r = pk_r$, and $mk' = mk$.
3. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input mk', pk'_r, and $param'$. $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state'$.
4. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, a release time t^*, and the state information $state = state'$.

The challenger will then randomly choose a bit $b \in \{0, 1\}$ and compute the challenge TRE-PC encryption $(C^*, V_{C^*}) = Enc(m_b, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C^*, the pre-open key V_{C^*}, and the state information $state$.
2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*||0, V_{C^*}, state)$. $\mathcal{B}_2$ eventually terminates by outputting a bit b'.
3. $\mathcal{A}_2$ terminates by outputting the bit b'.

It is clear to see that A provides a perfect environment in which B can run, $\mathcal{A}$ is a legitimate IND-TR-CPA$_{\mathrm{TS}}$ attacker, and $\mathcal{A}$'s advantage equals to δ. Since E is an IND-TR-CPA$_{\mathrm{TS}}$ TRE-PC scheme, then δ should be negligible, which proves the lemma. □

Lemma 2. *E' is not IND-TR-CCA$_{OS}$ secure.*

Proof. To prove the claim, we only need to show that the an IND-TR-CCA$_{\mathrm{OS}}$ attacker has non-negligible advantage in attacking E'. The attack is quite easy: On receiving the challenge $(C^*||0, V_{C^*})$ from the challenger, $\mathcal{A}_2$ makes a query to the oracle $\mathsf{Dec}_{\mathsf{PK}}{}'$ on the input $(C^*||1, t^*)$. It is easy to see that the oracle $\mathsf{Dec}_{\mathsf{PK}}{}'$ returns m_b, which allows the attacker to recover the bit b with the probability 1. □

As a result, we have proved the theorem. □

Theorem 2. *If there exists an IND-TR-CCA$_{OS}$ secure TRE-PC scheme E, then there exists a TRE-PC scheme E' that is IND-TR-CCA$_{OS}$ secure but not IND-TR-CPA$_{TS}$ secure.*

Proof. Suppose that $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ is an IND-TR-CCA$_{\mathrm{OS}}$ secure TRE-PC scheme and $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an IND-CPA secure public key encryption scheme. Consider the TRE-PC scheme E' where the algorithms are defined as follows.

1. The Setup' algorithm takes a security parameter 1^ℓ as input, and computes $(param, mk) = \mathsf{Setup}(1^\ell)$ and $(pk, sk) = \mathcal{K}(1^\ell)$. The public parameters are defined to be $param' = (param, pk)$. The master key is defined to be $mk' = (mk, sk)$.

2. The $\mathsf{Gen}'_{\mathsf{U}}$ algorithm takes a security parameter 1^ℓ as input, and computes $(pk'_r, sk'_r) = \mathsf{Gen_U}(1^\ell)$.

3. The $\mathsf{Ext}'_{\mathsf{TS}}$ algorithm take as input mk' and a release time t, and returns $\mathsf{Ext_{TS}}(mk, t)$.

4. The Enc' algorithm takes as input a message m, a release time t and the receiver's public key pk'_r, and returns a ciphertext $C = (C_1, C_2)$ and a pre-open key V_C, where:

$$\begin{aligned} C_1 &= \mathcal{E}(pk, m) \\ (C_2, V_C) &= \mathsf{Enc}(C_1||m, t, pk'_r) \end{aligned}$$

   We assume that C_1 is drawn from a prefix free set (such as a set of strings of a fixed length) so that it may be recovered from the arbitrary bitstring $C_1||m$.

5. The $\mathsf{Dec}'_{\mathsf{RK}}$ algorithm takes as input a ciphertext $C = (C_1, C_2)$, a pre-open key V_C, and a private key sk'_r, and computes

$$C'_1||m = \mathsf{Dec_{RK}}(C_2, V_C, sk'_r)\,.$$

   If $C'_1 \neq C_1$, then the algorithm outputs $\perp$. Otherwise it returns m.

6. The $\mathsf{Dec}'_{\mathsf{PK}}$ algorithm takes as input a ciphertext $C = (C_1, C_2)$, a timestamp TS_t, and a private key sk'_r, and computes

$$C'_1||m = \mathsf{Dec_{PK}}(C_2, TS_t, sk'_r)\,.$$

   If $C'_1 \neq C_1$, then the algorithm outputs $\perp$. Otherwise it returns m.

We begin by showing that E' is IND-TR-CCA$_{\mathrm{OS}}$ secure. Consider the following game played between an attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ and a hypothetical challenger. The game is parameterised by two bits $(b_1, b_2) \in \{0,1\}^2$ and a security parameter 1^ℓ, and runs as follows:

1. Game Setup: The challenger runs Setup' to generate public parameters $param' = (param, pk)$ and a private key $mk' = (mk, sk)$. The challenger also runs $\mathsf{Gen}'_{\mathsf{U}}$ to generate a public/private key pair (pk'_r, sk'_r).

2. Phase 1: The attacker executes $\mathcal{B}_1$ on the input $(pk'_r, param')$. $\mathcal{B}_1$ has access to the following oracles:

   - $\mathsf{Ext}'_{\mathsf{TS}}$ oracle, which takes as input a release time t and returns $\mathsf{Ext}'_{\mathsf{TS}}(mk, t)$.
   - $\mathsf{Dec}'_{\mathsf{RK}}$ oracle, which takes as input a ciphertext C and a pre-open key V_C, and returns $\mathsf{Dec}'_{\mathsf{RK}}(C, V_C, sk'_r)$.
   - $\mathsf{Dec}'_{\mathsf{PK}}$ oracle, which takes as input a ciphertext C and a release time t, and returns $\mathsf{Dec}'_{\mathsf{PK}}(C, TS_t, sk'_r)$.

   $\mathcal{B}_1$ terminates by outputting two equal length message m_0 and m_1, a release time t^* and some state information *state*.

3. Challenge: The challenger computes $C_1^* = \mathcal{E}(pk, m_{b_1})$ and $(C_2^*, V_{C^*}) = \mathsf{Enc}(C_1^* || m_{b_2}, t^*, pk'_r)$. The challenge ciphertext is the pair $C^* = (C_1^*, C_2^*)$. The challenge pre-open key is V_{C^*}.

4. Phase 2: The attacker executes $\mathcal{B}_2$ on the input $(C^*, V_{C^*}, state)$. $\mathcal{B}_2$ has access to the same oracles as in Phase 1; however, $\mathcal{B}_2$ may not query the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle on the input (C^*, V_{C^*}) or the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle on the input (C^*, t^*). $\mathcal{B}_2$ terminates by outputting a guessing bit b for b_2.

Let $Exp(b_1, b_2)$ be the event that $\mathcal{B}$ outputs 1 in the above game.

Lemma 3. *$\mathcal{B}$'s advantage in winning the IND-TR-CCA$_{OS}$ game is equal to*

$$\frac{1}{2} \left| Pr[Exp(0,0)] - Pr[Exp(1,1)] \right|$$

Proof. In an IND-TR-CCA$_{\mathrm{OS}}$ attack game, the attacker's advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$, where b is the bit randomly selected by the challenger in constructing the challenge ciphertext, and b' is the bit output by the attacker at the end of the game. The probability $Pr[b = b']$ can be computed as:

$$\begin{aligned} Pr[b = b'] &= Pr[b = b'|b = 0]Pr[b = 0] + Pr[b = b'|b = 1]Pr[b = 1] \\ &= \frac{1}{2}Pr[b' = 0|b = 0] + \frac{1}{2}Pr[b' = 1|b = 1] \\ &= \frac{1}{2}(1 - Pr[b' = 1|b = 0] + Pr[b' = 1|b = 1]) \\ &= \frac{1}{2} + \frac{1}{2}(Pr[b' = 1|b = 1] - Pr[b' = 1|b = 0]) \end{aligned}$$

However, by inspection of the game which defines the event $Exp(b_1, b_2)$, it is clear that

$$Pr[b' = 1|b = 0] = Pr[Exp(0,0)] \text{ and } Pr[b' = 1|b = 1] = Pr[Exp(1,1)].$$

Therefore, the result holds.

□

We note, by the triangle inequality, that we have that $\mathcal{B}$'s advantage is bounded by

$$\frac{1}{2}\left|Pr[Exp(0,0)] - Pr[Exp(0,1)]\right| + \frac{1}{2}\left|Pr[Exp(0,1)] - Pr[Exp(1,1)]\right|$$

Lemma 4. *If E is an IND-TR-CCA$_{OS}$ secure TRE-PC scheme, then*

$$\left|Pr[Exp(0,0)] - Pr[Exp(0,1)]\right|$$

is negligible as a function of the security parameter 1^{ℓ}.

Proof. We show that there exists an IND-TR-CCA$_{\mathrm{OS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that has advantage

$$\frac{1}{2}\left|Pr[Exp(0,0)] - Pr[Exp(0,1)]\right| .$$

Hence, we will be able to conclude that $|Pr[Exp(0,0)] - Pr[Exp(0,1)]|$ is negligible as E is IND-TR-CCA$_{\mathrm{OS}}$ secure.

The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public parameters $param$ and the public key pk'_r.
2. $\mathcal{A}_1$ computes an encryption key pair $(pk, sk) = \mathcal{K}(1^{\ell})$.
3. $\mathcal{A}_1$ sets $param' = (param, pk)$.
4. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input pk'_r and $param'$.
   - If $\mathcal{B}_1$ makes a extraction oracle query for a time t, then $\mathcal{A}_1$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ queries the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}_1$ queries its $\mathsf{Dec}_{\mathsf{RK}}$ oracle on (C_2, V_C). It receives $C'_1||m$ from the oracle. If $C'_1 \neq C_1$ then $\mathcal{A}_1$ returns $\perp$ to $\mathcal{B}_1$. Otherwise $\mathcal{A}_1$ returns the message m to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ queries the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then $\mathcal{A}_1$ queries its $\mathsf{Dec}_{\mathsf{PK}}$ oracle on (C_2, t). It receives $C'_1||m$ from the oracle. If $C'_1 \neq C_1$ then $\mathcal{A}_1$ returns $\perp$ to $\mathcal{B}_1$. Otherwise $\mathcal{A}_1$ returns the message m to $\mathcal{B}_1$.

   $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, and some state information $state'$.
5. $\mathcal{A}_1$ computes $C^*_1 = \mathcal{E}(pk, m_0)$.
6. $\mathcal{A}_1$ terminates by outputting the messages $C^*_1||m_0$ and $C^*_1||m_1$, a release time t^* and the state information $state = (state', C^*_1, pk, sk)$.

The challenger will then randomly choose a bit $b_2 \in \{0,1\}$ and compute the challenge TRE-PC encryption $(C_2^*, V_{C^*}) = Enc(m_{b_2}, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C_2^*, the challenge pre-open key V_{C^*} and the state information $state = (state', C_1^*, pk, sk)$. It sets $C^* = (C_1^*, C_2^*)$.
2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, V_{C^*}, state')$.
   - If $\mathcal{B}_2$ makes a extraction oracle query for a time t, then $\mathcal{A}_2$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_2$.
   - If $\mathcal{B}_2$ queries the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then
     - If $C_2 = C_2^*$ and $V_C = V_{C^*}$ then $\mathcal{A}_2$ returns $\perp$ to $\mathcal{B}_2$.
     - Otherwise $\mathcal{A}_2$ queries its $\mathsf{Dec}_{\mathsf{RK}}$ oracle on (C_2, V_C). It receives $C_1'||m$ from the oracle. If $C_1' \neq C_1$ then $\mathcal{A}_2$ returns $\perp$ to $\mathcal{B}_2$. Otherwise $\mathcal{A}_2$ returns the message m to $\mathcal{B}_2$.
   - If $\mathcal{B}_2$ queries the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then
     - If $C_2 = C_2^*$ and $t = t^*$ then $\mathcal{A}_2$ returns $\perp$ to $\mathcal{B}_2$.
     - Otherwise $\mathcal{A}_2$ queries its $\mathsf{Dec}_{\mathsf{PK}}$ oracle on (C_2, t). It receives $C_1'||m$ from the oracle. If $C_1' \neq C_1$ then $\mathcal{A}_2$ returns $\perp$ to $\mathcal{B}_2$. Otherwise $\mathcal{A}_2$ returns the message m to $\mathcal{B}_2$.

   $\mathcal{B}_2$ eventually terminates by outputting a bit b'.
3. $\mathcal{A}_2$ terminates by outputting the bit b'.

It is clear to see that the decryption oracles that $\mathcal{A}$ provides perfectly simulate the decryption oracles for $\mathcal{B}$ providing that $\mathcal{A}_2$ does not incorrectly respond $\perp$ for a ciphertext (C_1, C_2). There are two case in which this might occur:

- If $\mathcal{B}_2$ makes a $\mathsf{Dec}_{\mathsf{RK}}$ query on a ciphertext (C_1, C_2) with $C_1 \neq C_1^*$, $C_2 = C_2^*$ and $V_C = V_{C^*}$.
- If $\mathcal{B}_2$ makes a $\mathsf{Dec}_{\mathsf{PK}}$ query on a ciphertext (C_1, C_2) with $C_1 \neq C_1^*$, $C_2 = C_2^*$ and $t = t^*$.

In either case, we would recover $C_1^*||m_{b_2}$ after we apply the TRE-PC decryption algorithm. Hence, the ciphertext is invalid because $C_1 \neq C_1^*$. Hence, the response of $\perp$ is correct.

Therefore, $\mathcal{A}$ is a legitimate IND-TR-CCA$_{\mathrm{OS}}$ attacker. Its advantage can be defined as

$$\frac{1}{2} \left| Pr[\mathcal{A} \text{ outputs } 1 | b_2 = 0] - Pr[\mathcal{A} \text{ outputs } 1 | b_2 = 1] \right|,$$

and so this value is negligible as E is an IND-TR-CCA$_{\mathrm{OS}}$ TRE-PC scheme. However, this value is equal to

$$\frac{1}{2}\left|Pr[Exp(0,0)] - Pr[Exp(0,1)]\right|$$

which proves the lemma. □

Lemma 5. *If $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an IND-CPA secure encryption scheme, then*

$$\left|Pr[Exp(0,1)] - Pr[Exp(1,1)]\right|$$

is negligible as a function of the security parameter 1^ℓ.

Proof. We show that there exists an IND-CPA attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, which makes use of $\mathcal{B}$ as a subroutine, that has advantage

$$\frac{1}{2}\left|Pr[Exp(0,1)] - Pr[Exp(1,1)]\right|.$$

Hence, we conclude that $|Pr[Exp(0,1)] - Pr[Exp(1,1)]|$ is negligible as $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is IND-CPA secure.

The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public key pk from the challenger.
2. $\mathcal{A}_1$ computes $(param, mk) = \mathsf{Setup}(1^\ell)$ and $(pk'_r, sk'_r) = \mathsf{Gen}'_{\mathsf{U}}(1^\ell)$.
3. $\mathcal{A}_1$ sets $param' = (param, pk)$.
4. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input of pk'_r and $param'$.
   - If $\mathcal{B}_1$ makes an $\mathsf{Ext}'_{\mathsf{TS}}$ query for a time t, then $\mathcal{A}_1$ computes $TS_t = \mathsf{Ext}_{\mathsf{TS}}(mk, t)$ and returns TS_t to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ makes a $\mathsf{Dec}_{\mathsf{RK}}$ query for a ciphertext $C = (C_1, C_2)$ and a pre-open key V_C, then $\mathcal{A}_1$ computes $C'_1||m = \mathsf{Dec}_{\mathsf{RK}}(C_2, V_C, sk_r)$. If $C'_1 \neq C_1$ then $\mathcal{A}_1$ returns $\perp$; otherwise $\mathcal{A}_1$ returns m to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ makes a $\mathsf{Dec}_{\mathsf{PK}}$ query for a ciphertext $C = (C_1, C_2)$ and a release time t, then $\mathcal{A}_1$ computes $C'_1||m = \mathsf{Dec}_{\mathsf{PK}}(C_2, TS_t, sk_r)$. If $C'_1 \neq C_1$ then $\mathcal{A}_1$ returns $\perp$; otherwise $\mathcal{A}_1$ returns m to $\mathcal{B}_1$.

   $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^* and some state information $state'$.
5. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, and some state information $state = (state', pk, param, mk, pk_r, sk_r, m_0, m_1, t^*)$.

The challenger will then randomly choose a bit $b_1 \in \{0,1\}$ and compute the challenge public-key encryption $C_1^* = \mathcal{E}(m_{b_1})$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C_1^* and the state information $state = (state', pk, param, mk, pk_r, sk_r, m_0, m_1, t^*)$.

2. $\mathcal{A}_2$ computes $(C_2^*, V_{C^*}) = \mathsf{Enc}(C_1^*||m_1, t^*, pk_r)$ and sets $C^* = (C_1^*, C_2^*)$.

3. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, V_{C^*}, state')$. If $\mathcal{B}_2$ makes any oracle queries, then $\mathcal{A}_2$ answers them in exactly the same way as $\mathcal{A}_1$ would have done. $\mathcal{B}_2$ terminates by outputting a bit b'.

4. $\mathcal{A}_2$ terminates by outputting the bit b'.

Clearly, for the same reasons as in the previous lemma, we have that $\mathcal{A}$'s advantage is equal to $\frac{1}{2}\left|Pr[Exp(0,1)] - Exp(1,1)\right|$ and hence this value is negligible. □

This proves that E' is IND-TR-CCA$_{\mathrm{OS}}$ secure. Hence, it only remains to show that E' is not IND-TR-CPA$_{\mathrm{TS}}$ secure.

Lemma 6. *E' is not IND-TR-CPA$_{TS}$ secure.*

Proof. To prove the claim, we only need to show that the an IND-TR-CPA$_{\mathrm{TS}}$ attacker has non-negligible advantage in attacking E'. The attack is quite easy: Suppose that the challenge is (C^*, V_{C^*}), where $C^* = (C_1^*, C_2^*)$. Since the attacker has access to sk, then it can immediately compute $m_b = \mathcal{D}_{sk}(C_1^*)$, which means it can succeed in guessing b with the probability 1. □As a result, the theorem gets proved. □

4.2 Relation between IND-TR-CCA$_{\mathrm{TS}}$ and IND-TR-CPA$_{\mathrm{IS}}$

We prove that IND-TR-CCA$_{\mathrm{TS}} \not\rightarrow$ IND-TR-CPA$_{\mathrm{IS}}$ by the following theorem.

Theorem 3. *If there exists an IND-TR-CCA$_{TS}$ secure TRE-PC scheme E, then there exists a TRE-PC scheme E' which is IND-TR-CCA$_{TS}$ secure but not IND-TR-CPA$_{IS}$ secure.*

Proof. Suppose that $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ is an IND-TR-CCA$_{\mathrm{TS}}$ secure TRE-PC scheme and $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an IND-CPA secure public key encryption scheme. Consider the TRE-PC scheme E' where the algorithms are defined as follows.

1. The Setup' algorithm takes a security parameter 1^ℓ as input, and computes the public/private parameters $(param', mk') = \mathsf{Setup}(1^\ell)$.

2. The $\mathsf{Gen}'_{\mathsf{U}}$ algorithm takes as input the public parameters $param'$, and computes $(pk_r, sk_r) = \mathsf{Gen_U}(1^\ell)$ and and $(pk, sk) = \mathcal{K}(1^\ell)$. The user's public key is defined to be $pk'_r = (pk_r, pk)$. The user's private key is defined to be $sk'_r = (sk_r, sk)$.

3. The $\mathsf{Ext}'_{\mathsf{TS}}$ algorithm take as input mk', and a release time t, and returns $\mathsf{Ext_{TS}}(mk', t)$.

4. The Enc' algorithm takes as input a message m, a release time t and the receiver's public key pk'_r, and returns a ciphertext $C = (C_1, C_2)$ and a pre-open key V_C, where:

$$\begin{aligned} C_1 &= \mathcal{E}(pk, m) \\ (C_2, V_C) &= \mathsf{Enc}(C_1||m, t, pk_r) \end{aligned}$$

We assume that C_1 is drawn from a prefix free set (such as a set of strings of a fixed length) so that it may be recovered from the arbitrary bitstring $C_1||m$.

5. The $\mathsf{Dec}'_{\mathsf{RK}}$ algorithm takes as input a ciphertext $C = (C_1, C_2)$, a pre-open key V_C, and a private key sk'_r, and computes

$$C'_1||m = \mathsf{Dec}_{\mathsf{RK}}(C_2, V_C, sk_r)\,.$$

If $C'_1 \neq C_1$, then the algorithm outputs $\perp$. Otherwise it returns m.

6. The $\mathsf{Dec}'_{\mathsf{PK}}$ algorithm takes as input a ciphertext $C = (C_1, C_2)$, a timestamp TS_t, and a private key sk'_r, and computes

$$C'_1||m = \mathsf{Dec}_{\mathsf{PK}}(C_2, TS_t, sk_r)\,.$$

If $C'_1 \neq C_1$, then the algorithm outputs $\perp$. Otherwise it returns m.

It is easy to see that E' is constructed in the same way as in the proof of Theorem 2, except that (pk, sk) is possessed by the receiver instead of by the trusted time server. Therefore, following exactly the same procedure as in the proof of Theorem 2, we can prove that E' is IND-TR-CCA$_{\mathrm{TS}}$ secure. Hence, it only remains to show that E' is not IND-TR-CPA$_{\mathrm{IS}}$ secure.

Lemma 7. *E' is not IND-TR-CPA$_{IS}$ secure.*

Proof. To prove the claim, we only need to show that the an IND-TR-CPA$_{\mathrm{TS}}$ attacker has non-negligible advantage in attacking E'. The attack is quite easy: Suppose that the challenge is $C^* = (C_1^*, C_2^*)$. Since the attacker has access to sk, then it can immediately compute $m_b = \mathcal{D}_{sk}(C_1^*)$, which means it can succeed in guessing b with the probability 1. □As a result, the theorem gets proved. □

4.3 Relation between IND-TR-CPA$_{\mathbf{IS}}$ and IND-TR-CPA$_{\mathbf{OS}}$

We prove that IND-TR-CPA$_{\mathrm{IS}} \nrightarrow$ IND-TR-CPA$_{\mathrm{OS}}$ by the following theorem.

Theorem 4. *If there exists an IND-TR-CPA$_{IS}$ secure TRE-PC scheme E, then there exists a TRE-PC scheme E' which is IND-TR-CPA$_{IS}$ secure but not IND-TR-CPA$_{OS}$ secure.*

Proof. Suppose that $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ is an IND-TR-CPA$_{\mathrm{IS}}$ secure TRE-PC scheme. Consider the TRE-PC scheme E' where the algorithms are defined as follows.

1. The algorithms $\mathsf{Setup}', \mathsf{Gen}'_{\mathsf{U}}, \mathsf{Ext}'_{\mathsf{TS}}, \mathsf{Dec}'_{\mathsf{PK}}$ are defined in the same way as in E.

2. The Enc' algorithm takes a message m, a release time t and the receiver's public key pk'_r as input, and returns a ciphertext C and a pre-open key V_C, where:

$$\begin{aligned} (C, V'_C) &= \mathsf{Enc}(m, t, pk'_r) \\ V_C &= V'_C||m \end{aligned}$$

   We assume that V'_C is drawn from a prefix free set (such as a set of strings of a fixed length) so that it may be recovered from the arbitrary bitstring $V'_C||m$.

3. The $\mathsf{Dec}'_{\mathsf{RK}}$ algorithm takes a ciphertext C, a pre-open key $V'_C||m$, and a private key sk'_r as input, and returns $\mathsf{Dec_{RK}}(C, V'_C, sk'_r)$.

The validity of this theorem lies in the following two lemmas.

Lemma 8. *E' is IND-TR-CPA$_{IS}$ secure.*

Proof. Suppose an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ has the advantage δ in attacking E'. We show that there exists an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that also has advantage δ. Hence, we will be able to conclude that δ is negligible as E is IND-TR-CPA$_{\mathrm{IS}}$ secure.

The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public parameters $param$ and the public/private key pair (pk_r, sk_r).

2. $\mathcal{A}_1$ sets $param' = param$, $pk'_r = pk_r$, and $sk'_r = sk$.

3. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input pk'_r, sk'_r, and $param'$. If $\mathcal{B}_1$ makes a extraction oracle query for a time t, then $\mathcal{A}_1$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_1$. $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state'$.

4. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, a release time t^*, and the state information $state = state'$.

The challenger will then randomly choose a bit $b \in \{0, 1\}$ and compute the challenge TRE-PC encryption C^*, where $(C^*, V_{C^*}) = Enc(m_b, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C^* and the state information *state*.

2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, state)$. If $\mathcal{B}_2$ makes a extraction oracle query for a time $t < t^*$, then $\mathcal{A}_2$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_2$. $\mathcal{B}_2$ eventually terminates by outputting a bit b'.

3. $\mathcal{A}_2$ terminates by outputting the bit b'.

It is clear to see that $\mathcal{A}$ provides perfect simulation for the oracles for $\mathcal{B}$ may query, $\mathcal{A}$ is a legitimate IND-TR-CPA$_{\mathrm{IS}}$ attacker, and $\mathcal{A}$'s advantage equals to δ. Since E is an IND-TR-CPA$_{\mathrm{IS}}$ TRE-PC scheme, then δ should be negligible, which proves the lemma. □

Lemma 9. *E' is not IND-TR-CPA$_{OS}$ secure.*

Proof. It is clear to see that an IND-TR-CPA$_{\mathrm{OS}}$ attacker can identify the random bit chosen by the challenger with the probability 1, because the pre-open key always contains the plaintext. As a result, this lemma is valid. □As a result, the theorem gets proved. □

4.4 Relation between IND-TR-CCA$_{\mathrm{TS}}$ and Binding

We prove that "IND-TR-CCA$_{\mathrm{TS}}$ $\not\longrightarrow$ Binding" by the following theorem.

Theorem 5. *If there exists an IND-TR-CCA$_{TS}$ secure TRE-PC scheme E, then there exists a TRE-PC scheme E' which is IND-TR-CCA$_{TS}$ secure but not binding.*

Proof. Suppose that $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ is an IND-TR-CCA$_{\mathrm{TS}}$ secure TRE-PC scheme. Consider the TRE-PC scheme E' where the algorithms are defined as follows.

1. The algorithms $\mathsf{Setup}', \mathsf{Gen}'_{\mathsf{U}}, \mathsf{Ext}'_{\mathsf{TS}}, \mathsf{Enc}'$ are defined in the same way as in E.

2. The algorithm $\mathsf{Dec}'_{\mathsf{RK}}$ is defined in the same way as $\mathsf{Dec_{RK}}$, except that it returns a random message from the plaintext space when $\mathsf{Dec_{RK}}$ returns $\perp$.

3. The algorithm $\mathsf{Dec}'_{\mathsf{PK}}$ is defined in the same way as $\mathsf{Dec_{PK}}$, except that it returns a random message from the plaintext space when $\mathsf{Dec_{PK}}$ returns $\perp$.

The validity of this theorem lies in the following two lemmas.

Lemma 10. *E' is IND-TR-CCA$_{TS}$ secure.*

Proof. Suppose an IND-TR-CCA$_{\mathrm{TS}}$ attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ has the advantage δ in attacking E'. We show that there exists an IND-TR-CCA$_{\mathrm{TS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that also has advantage δ. Hence, we will be able to conclude that δ is negligible as E is IND-TR-CCA$_{\mathrm{TS}}$ secure.

The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public parameters $param$, the public key pk_r, and the master key mk.
2. $\mathcal{A}_1$ sets $param' = param$, $pk'_r = pk_r$, and $mk' = mk$.
3. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input of mk', pk'_r, and $param'$.
   - If $\mathcal{B}_1$ queries the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle with the ciphertext C and the pre-open key V_C, then $\mathcal{A}_1$ queries its $\mathsf{Dec}_{\mathsf{RK}}$ oracle on (C, V_C). It receives m' from the oracle. If $m' = \perp$ then $\mathcal{A}_1$ returns a random message from the plaintext space to $\mathcal{B}_1$. Otherwise $\mathcal{A}_1$ returns the message m' to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ queries the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle with the ciphertext C and for the time t, then $\mathcal{A}_1$ queries its $\mathsf{Dec}_{\mathsf{PK}}$ oracle on (C, t). It receives m' from the oracle. If $m' = \perp$ then $\mathcal{A}_1$ returns a random message from the plaintext space to $\mathcal{B}_1$. Otherwise $\mathcal{A}_1$ returns the message m' to $\mathcal{B}_1$.

   $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state'$.
4. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, a release time t^*, and the state information $state = state'$.

The challenger will then randomly choose a bit $b \in \{0, 1\}$ and compute the challenge TRE-PC encryption $(C^*, V_{C^*}) = Enc(m_b, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C^*, the challenge pre-open key V_{C^*} and the state information $state$.
2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, V_{C^*}, state)$.
   - If $\mathcal{B}_2$ queries the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}_2$ queries its $\mathsf{Dec}_{\mathsf{RK}}$ oracle on (C, V_C). Suppose $\mathcal{A}_2$ receives m' as the response. If $m' = \perp$ then $\mathcal{A}_2$ returns a random message from the plaintext space to $\mathcal{B}_2$. Otherwise $\mathcal{A}_2$ returns the message m' to $\mathcal{B}_2$.
   - If $\mathcal{B}_2$ queries the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then $\mathcal{A}_2$ queries its $\mathsf{Dec}_{\mathsf{PK}}$ oracle on (C, t). Suppose $\mathcal{A}_2$ receives m' as the response. If $m' = \perp$ then $\mathcal{A}_2$ returns a random message from the plaintext space to $\mathcal{B}_2$. Otherwise $\mathcal{A}_2$ returns the message m' to $\mathcal{B}_2$.

$\mathcal{B}_2$ eventually terminates by outputting a bit b'.

3. $\mathcal{A}_2$ terminates by outputting the bit b'.

It is clear to see that $\mathcal{A}$ provides perfect simulation for the oracles that $\mathcal{B}$ may query, $\mathcal{A}$ is a legitimate IND-TR-CCA$_{\mathrm{TS}}$ attacker, and $\mathcal{A}$'s advantage equals to δ. Since E is an IND-TR-CCA$_{\mathrm{TS}}$ secure TRE-PC scheme, then δ should be negligible, which proves the lemma. □

Lemma 11. *E' is not binding.*

Proof. We first construct a two-stage algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and then construct a binding attacker $\mathcal{A}'$ for E'. The sub-algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ are defined as follows:

1. $\mathcal{A}_1$ takes $(pk_r, param)$ as input, where pk_r is a string from the same space as pk'_r and $param$ is a string from the same space as $param'$, and returns (m_0, m_1, t^*), where m_0 and m_1 are two equal length messages randomly chosen from the plaintext space of E' and $t^* = 2$.

2. $\mathcal{A}_2$ takes (m_0, m_1, C, V_C) as input, where C is from the ciphertext space of E', V_C is from the pre-open key space of E'. $\mathcal{A}_2$ sets $t^\dagger = 1$ and makes a $\mathsf{Dec}'_{\mathsf{PK}}$ query on the input of $(C, t^\dagger)$. If $\mathcal{A}_2$ receives $m^\dagger$ as the response, then it terminates by outputting a bit b' which is defined as follows: $b' = i$ if $m^\dagger = m_i$ $(0 \leq i \leq 1)$; otherwise, b' is set to be a random bit.

The attacker $\mathcal{A}'$ is defined as follows. $\mathcal{A}'$ receives $(pk'_r, param')$ from the challenger, then

1. Run $\mathcal{A}_1$ on the input of $(pk'_r, param')$ and get the output (m_0, m_1, t^*),
2. Select a random bit $b \in \{0, 1\}$ and compute $(C^*, V_{C^*}) = \mathsf{Enc}(m_b, t^*, pk'_r)$,
3. Run $\mathcal{A}_2$ on the input of (m_0, m_1, C^*, V_{C^*}),
4. On receiving $\mathcal{A}_2$'s $\mathsf{Dec}'_{\mathsf{PK}}$ query, make a $\mathsf{Dec}'_{\mathsf{PK}}$ query to its own oracle on the input of $(C^*, t^\dagger)$, return the output to $\mathcal{A}_2$,
5. Get the output b' from $\mathcal{A}_2$ and terminate by outputting $(C^*, V_{C^*}, t^\dagger)$.

From the description, it is clear that $\mathcal{A}'$ is a legitimate binding attacker, $\mathcal{A}'$'s advantage can be denoted as $\delta = Pr[m^\dagger \neq m_b]$, and $|Pr[b = b'] - \frac{1}{2}|$ is negligible because E' is also IND-TR-CCA$_{\mathrm{TS}}$ secure.

Let E_1 and E_2 be the events that $m^\dagger = m_b$ and $m^\dagger = m_{\bar{b}}$ respectively, where $\bar{b} = |b - 1|$. Let E_3 be the event that neither E_1 nor E_2 occurs. The probability $Pr[b = b']$ can be denoted as follows:

$$
\begin{aligned}
Pr[b = b'] &= \sum_{i=1}^{3} Pr[E_i] Pr[b = b' | E_i] && (1)\\
&= Pr[E_1] + \frac{1}{2} Pr[E_3] && (2)\\
&= 1 - Pr[E_2] - \frac{1}{2} Pr[E_3] && (3)
\end{aligned}
$$

Note that the reduction from (1) to (2) is based on the fact that $Pr[b = b'|E_1] = 1, Pr[b = b'|E_2] = 0, Pr[b = b'|E_3] = \frac{1}{2}$, while the reduction from (2) to (3) is based on the fact that $\sum_{i=1}^{3} Pr[E_i] = 1$.

As a result, $|Pr[b = b'] - \frac{1}{2}|$ is negligible implies that $|\frac{1}{2} - \delta'|$, where $\delta' = Pr[E_2] + \frac{1}{2}Pr[E_3]$, is negligible, and therefore δ' is non-negligible. Since $\delta = Pr[E_2] + Pr[E_3] \geq Pr[E_2] + \frac{1}{2}Pr[E_3] = \delta'$, then it is clear that δ is also non-negligible and the lemma is valid. □

Combining the results in the previous lemmas, the theorem gets proved. □

4.5 Relation between Binding and IND-TR-CPA$_{\mathrm{OS}}$

We prove that "Binding $\not\longrightarrow$ IND-TR-CPA$_{\mathrm{OS}}$" by the following theorem.

Theorem 6. *There exists a TRE-PC scheme E which is binding but not IND-TR-CPA$_{OS}$ secure.*

Proof. Consider the following TRE-PC scheme E where the algorithms are defined as follows.

1. The Setup algorithm takes a security parameter 1^ℓ as input, and sets the master key $mk = 1$ and the public parameters $param = 1$.
2. The $\mathsf{Gen_U}$ algorithm takes a security parameter 1^ℓ as input, and sets a public/private key pair (pk_r, sk_r), where $pk_r = sk_r = 1$. The plaintext and ciphertext space is denoted as $\{0,1\}^k$
3. The $\mathsf{Ext_{TS}}$ algorithm take mk and a release time t as input, and returns $TS_t = 1$.
4. The Enc algorithm takes a message m, a release time t and the receiver's public key pk_r as input, and returns a ciphertext $C = m$ and a pre-open key $V_C = 1$.
5. The $\mathsf{Dec_{RK}}$ algorithm takes a ciphertext C, a pre-open key V_C, and a private key sk_r as input, and returns $m = C$.
6. The $\mathsf{Dec_{PK}}$ algorithm takes as input a ciphertext C, a timestamp TS_t, and a private key sk_r, and returns $m = C$.

From the description, it is easy to see that E is binding but not not IND-TR-CPA$_{\mathrm{OS}}$ secure. As a result, the theorem gets proved. □

It is clear that this scheme is also not IND-TR-CPA$_{\mathrm{IS}}$ secure.

4.6 Relations between IND-TR-CPA$_{\mathrm{IS}}$ and Binding

We prove that "IND-TR-CPA$_{\mathrm{IS}}$ $\not\longrightarrow$ Binding" by the following theorem.

Theorem 7. *If there exists an IND-TR-CPA$_{IS}$ secure TRE-PC scheme E, then there exists a TRE-PC scheme E' which is IND-TR-CPA$_{IS}$ secure but not binding.*

Proof. Suppose that $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ is an IND-TR-CPA$_{\mathrm{IS}}$ secure TRE-PC scheme. Consider the TRE-PC scheme E' where the algorithms are defined as follows.

1. The algorithms $\mathsf{Setup}', \mathsf{Gen}'_{\mathsf{U}}, \mathsf{Ext}'_{\mathsf{TS}}, \mathsf{Enc}'$ are defined in the same way as in E.
2. The algorithm $\mathsf{Dec}'_{\mathsf{RK}}$ is defined in the same way as $\mathsf{Dec_{RK}}$, except that it returns a random message from the plaintext space when $\mathsf{Dec_{RK}}$ returns $\perp$.
3. The algorithm $\mathsf{Dec}'_{\mathsf{PK}}$ is defined in the same way as $\mathsf{Dec_{PK}}$, except that it returns a random message from the plaintext space when $\mathsf{Dec_{PK}}$ returns $\perp$.

The validity of this theorem lies in the following two lemmas.

Lemma 12. *E' is IND-TR-CPA$_{IS}$ secure.*

Proof. Suppose an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ has the advantage δ in attacking E'. We show that there exists an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that also has advantage δ. Hence, we will be able to conclude that δ is negligible as E is IND-TR-CPA$_{\mathrm{IS}}$ secure.

The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public parameters $param$, the public key pk_r, and the private key sk_r.
2. $\mathcal{A}_1$ sets $param' = param$, $pk'_r = pk_r$, and $sk'_r = sk_r$.
3. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input of pk'_r, sk'_r, and $param'$. If $\mathcal{B}_1$ makes a extraction oracle query for a time t, then $\mathcal{A}_1$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_1$. $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state'$.
4. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, a release time t^*, and the state information $state = state'$.

The challenger will then randomly choose a bit $b \in \{0, 1\}$ and compute the challenge TRE-PC encryption C^*, where $(C^*, V_{C^*}) = Enc(m_b, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C^* and the state information $state$.
2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, state)$. If $\mathcal{B}_2$ makes a extraction oracle query for a time $t < t^*$, then $\mathcal{A}_2$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_2$. $\mathcal{B}_2$ eventually terminates by outputting a bit b'.

3. $\mathcal{A}_2$ terminates by outputting the bit b'.

It is clear to see that $\mathcal{A}$ provides perfect simulation for the oracles that $\mathcal{B}$ may query, $\mathcal{A}$ is a legitimate IND-TR-CPA$_{\mathrm{IS}}$ attacker, and $\mathcal{A}$'s advantage equals to δ. Since E is an IND-TR-CPA$_{\mathrm{IS}}$ secure TRE-PC scheme, then δ should be negligible, which proves the lemma. □

Lemma 13. *E' is not binding.*

Proof. The proof of this lemma is based on the fact that if E' is IND-TR-CPA$_{\mathrm{IS}}$ secure then a polynomial-time attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ only has negligible advantage in the following game.

1. Game setup: The challenger runs Setup' to generate the time server's master key mk' and the public system parameters $param'$. The challenger also runs $\mathsf{Gen}'_{\mathsf{U}}$ to generate a public/private key pair (pk'_r, sk'_r).
2. Phase 1: The attacker runs $\mathcal{B}_1$ on the input $(pk'_r, param')$. $\mathcal{B}_1$ has access to the following types of oracles:
   - $\mathsf{Ext}'_{\mathsf{TS}}$ oracle, which, on receiving a query for time t, returns $\mathsf{Ext}'_{\mathsf{TS}}(mk, t)$.
   - $\mathsf{Dec}'_{\mathsf{RK}}$ oracle, which, on receiving a query for (C, V_C), returns $\mathsf{Dec}'_{\mathsf{RK}}(C, V_C, sk'_r)$.
   - $\mathsf{Dec}'_{\mathsf{PK}}$ oracle, which, on receiving a query for (C, t), returns $\mathsf{Dec}'_{\mathsf{PK}}(C, TS_t, sk'_r)$.

   $\mathcal{B}_1$ terminates by outputting two equal length messages m_0, m_1 and a release time t^* which is larger than all the inputs to the $\mathsf{Ext}'_{\mathsf{TS}}$ oracle. In addition, $\mathcal{B}_1$ also outputs some state information *state*.
3. Challenge: The challenger picks a random bit $d \in \{0, 1\}$, computes $(C^*, V_{C^*}) = \mathsf{Enc}(m_d, t^*, pk'_r)$, and returns C^*.
4. Phase 2: The attacker runs $\mathcal{B}_2$ on the input $(C^*, state)$. $\mathcal{B}_2$ has access to the same types of oracles as $\mathcal{B}_1$. However, $\mathcal{B}_2$ may not make a $\mathsf{Ext}'_{\mathsf{TS}}$ query on a time $t \geq t^*$ or a $\mathsf{Dec}'_{\mathsf{PK}}$ query on the input (C, t), where $t \geq t^*$. $\mathcal{B}_2$ terminates by outputting a guessing bit $d' \in \{0, 1\}$.

In this attack game, instead of taking the private key sk'_r as an input as required by a legitimate IND-TR-CPA$_{\mathrm{IS}}$ attack game, the attacker is granted access to the decryption oracles. Since E' is IND-TR-CPA$_{\mathrm{IS}}$ secure, it is easy to see that the attacker's advantage in the above game (i.e. $|Pr[d' = d] - \frac{1}{2}|$) is negligible.

Then we can construct a binding attacker $\mathcal{A}'$ and an algorithm $\mathcal{A}$ identical to those in the proof of Lemma 11. The proof is also exactly the same as that of Lemma 11, except that in this case the probability $|Pr[b = b'] - \frac{1}{2}|$ is negligible because $\mathcal{B}$ only has negligible advantage in the above game. □

5 KEM, DEM, and TRE-PC KEM

In this section, we first review the security definitions for KEM and DEM, and then introduce concepts and security definitions for TRE-PC KEM which can be thought of as combining the functionality of a KEM and a TRE-PC scheme.

5.1 Preliminaries of KEM and DEM

A KEM consists of the following three algorithms:

- A probabilistic, polynomial-time key generation algorithm $\mathsf{KEM.Gen}$ that on input of a security parameter 1^ℓ ($\ell \geq 1$), outputs a public/private key pair (pk_r, sk_r).
- A probabilistic, polynomial-time encapsulation algorithm $\mathsf{KEM.Encap}$, which, on the input of a public key pk_r, outputs a pair (K, C), where K is a symmetric key and C is a ciphertext.
- A deterministic, polynomial-time decapsulation algorithm $\mathsf{KEM.Decap}$, which, on the input a ciphertext C and a private key sk_r, outputs either a symmetric key K or an error message $\perp$.

We assume that the range of possible keys K is some set of fixed length binary strings, $\{0,1\}^{\mathsf{KeyLen}(\ell)}$, where KeyLen is polynomial-time computable. The formal definitions for the securities of a KEM against an adaptive chosen ciphertext attack and a passive attack are as follows.

Definition 11. *A KEM is defined to be secure against an adaptive chosen ciphertext attack (IND-CCA2 secure), if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ only has negligible advantage in the following game.*

1. Game setup: The challenger runs $\mathsf{KEM.Gen}$ on the input of a security parameter 1^ℓ to generate a public/private key pair (pk_r, sk_r).
2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input of pk_r. During its execution, $\mathcal{A}_1$ has access to a decapsulation oracle, which, on the input of C, returns $\mathsf{KEM.Decap}(C, sk_r)$. $\mathcal{A}_1$ terminates by outputting some state information *state*.
3. Challenge: The challenger generates a challenge encapsulated pair as follows:
   (a) The challenger generates an encapsulated pair $(K_0, C^*) = \mathsf{KEM.Encap}(pk)$.
   (b) The challenger randomly selects $K_1 \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$.
   (c) The challenger randomly selects a bit $b \in \{0,1\}$, and returns (K_b, C^*).
4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input of $(K_b, C^*, state)$. During its execution, $\mathcal{A}_2$ has access to the decapsulation oracle. But $\mathcal{A}_2$ may not make a query on the input C^*. $\mathcal{A}_2$ terminates by outputting a guessing bit b'.

In this attack game, the attacker's advantage is defined to be $|Pr[b' = b] - \frac{1}{2}|$.

Definition 12. *A KEM is defined to be secure against a passive attack (IND-CPA secure) if it is IND-CCA2 secure against attackers that make no decapsulation queries.*

A DEM consists of the following two algorithms:

- A deterministic, polynomial-time encryption algorithm $\mathsf{DEM.Enc}$, which, on the input a message m and a symmetric key K, outputs a ciphertext C.
- A deterministic, polynomial-time decryption algorithm $\mathsf{DEM.Dec}$, which, on the input a ciphertext C and a symmetric key K, outputs a message m or an error message $\perp$.

We assume that the range of possible keys K is the same as that of the associated KEM, i.e., $\{0,1\}^{\mathsf{KeyLen}(\ell)}$. The formal definitions for the securities of a DEM against an adaptive chosen ciphertext attack and a passive attack are as follows.

Definition 13. *A DEM is defined to be secure against an adaptive chosen ciphertext attack (IND-CCA2 secure), if a two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ only has negligible advantage in the following game.*

1. Phase 1: The attacker runs $\mathcal{A}_1$. At some point, $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1. In addition, $\mathcal{A}_1$ also outputs some state information *state*.
2. Challenge: The challenger randomly selects a bit $b \in \{0,1\}$ and a symmetric key $K \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$, and returns $C^* = \mathsf{DEM.Enc}(m_b, K)$.
3. Phase 2: The attacker runs $\mathcal{A}_2$ on the input of $(C^*, state)$. During its execution, $\mathcal{A}_2$ has access to the decryption oracle, which, on the input of C, returns $\mathsf{DEM.Dec}(C, K)$. But $\mathcal{A}_2$ may not make a query on the input C^*. $\mathcal{A}_2$ terminates by outputting a guessing bit b'.

In this attack game, the attacker's advantage is defined to be $|Pr[b' = b] - \frac{1}{2}|$.

Definition 14. *A DEM is defined to be secure against a passive attack (IND-CPA secure) if it is IND-CCA2 secure against attackers that make no decapsulation query.*

5.2 Definition of TRE-PC KEM

In this subsection, we define TRE-PC KEM, which is a special type of KEM with timed-release and pre-open capabilities. A TRE-PC KEM consists of the following polynomial-time algorithms:

- TRE-PC-KEM.Setup: Run by the time server, this setup algorithm takes a security parameter 1^ℓ as input, and generates a secret master-key mk and the public parameters $param$. We assume that all subsequent algorithms take $param$ implicitly as an input
- $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$: Run by the time server, this timestamp extraction algorithm takes mk and a time t as input, and generates a timestamp TS_t.
- TRE-PC-KEM.Gen: Run by a user, this key generation algorithm takes a security parameter 1^ℓ as input, and outputs a public/private key pair (pk_r, sk_r).
- TRE-PC-KEM.Encap: Run by the message sender, this key encapsulation algorithm takes a release time t and a public key pk_r as input, and outputs (K, C, V_C), where K is a symmetric key, C is a ciphertext, V_C is the pre-open key of C.
- $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$: Run by the receiver, this decapsulation algorithm takes a ciphertext C, a pre-open key V_C, and the receiver's private key sk_r as input, and returns either the encapsulated key K or an error message $\perp$.
- $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$: Run by the receiver, this decryption algorithm takes a ciphertext C, a timestamp TS_t which is determined by the release time accompanied with C, and the receiver's private key sk_r as input, and returns either the encapsulated key K or an error message $\perp$.

It should be noted that $param$ is defaulted to be the input of all the algorithms except for TRE-PC-KEM.Setup.

5.3 Security definitions

For a TRE-PC KEM, we consider the same types of attackers as a TRE-PC scheme, and correspondingly we have the following security definitions.

5.4 Soundness of a TRE-PC KEM

Informally, the decapsulation algorithms of a sound TRE-PC KEM should always "undo" the output of the encapsulation algorithm. Formally, soundness is defined as follows.

Definition 15. *A TRE-PC KEM is sound if, for any time t and (K, C, V_C) where*

$$(K, C, V_C) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t, pk_r),$$

the following two requirements are satisfied

$$K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C, V_C, sk_r),$$

$$K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, TS_t, sk_r).$$

5.5 Binding of a TRE-PC KEM

Analogously to the definition given in 3.2, we give the following definition of the *binding* property.

Definition 16. *A TRE-PC KEM is binding if any polynomial-time attacker $\mathcal{A}$ has only a negligible probability of winning the following game.*

1. Game setup: The challenger runs $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ to generate a public/private key pair (pk_r, sk_r).

2. Challenge: The attacker $\mathcal{A}$ executes with input $(pk_r, param)$. At some point, $\mathcal{A}$ generates a ciphertext C^* for release at time t^* and a pre-open key V_{C^*}, and then terminates by outputting (C^*, t^*, V_{C^*}). During its execution, $\mathcal{A}$ has access to the following oracles:

   - An oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Ext_{TS}}$, which, on receiving a query for time t, returns $\mathsf{TRE\text{-}PC\text{-}KEM.Ext_{TS}}(mk, t)$.
   - An oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Decap_{RK}}$, which, on receiving a query for (C, V'_C), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap_{RK}}(C, V'_C, sk_r)$. Note that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key for C.
   - An oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Decap_{PK}}$, which, on receiving a query for (C, t'), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap_{PK}}(C, TS_{t'}, sk_r)$. Note that t' may not be the release time for C.

In this game, $\mathcal{A}$ wins if $O_1 \neq \perp$, $O_2 \neq \perp$, and $O_1 \neq O_2$, where

$$O_1 = \mathsf{TRE\text{-}PC\text{-}KEM.Decap_{RK}}(C^*, V_{C^*}, sk_r),$$

$$O_2 = \mathsf{TRE\text{-}PC\text{-}KEM.Decap_{PK}}(C^*, TS_{t^*}, sk_r).$$

5.5.1 Security against malicious outsiders

We define the securities of a TRE-PC KEM against outside attackers which do not know the time server's master key. Specifically, we define the security under an adaptive chosen ciphertext attack (IND-TR-KEM-CCA$_{\mathrm{OS}}$ security) and the security under an adaptive chosen plaintext attack (IND-TR-KEM-CPA$_{\mathrm{OS}}$ security).

Definition 17. *A TRE-PC KEM is IND-TR-KEM-CCA$_{OS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ only has negligible advantage in the following game.*

1. Game setup: The challenger runs $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input $(pk_r, param)$. $\mathcal{A}_1$ has access to the following oracles.

   - $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ oracle, which, on receiving a query for time t, returns $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}(mk, t)$.
   - $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ oracle, which, on receiving a query for (C, V'_C), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C, V'_C, sk_r)$. It should be noted that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key of C.
   - $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ oracle, which, on receiving a query for (C, t'), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, TS_{t'}, sk_r)$. It should be noted that t' need not be the legitimate release time for C.

   $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information *state*.

3. Challenge: The challenger generates the challenge as follows:

   (a) The challenger computes $(K_0, C^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$.
   (b) The challenger randomly selects $K_1 \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$.
   (c) The challenger randomly selects a bit $b \in \{0,1\}$, and returns (K_b, C^*, V_{C^*}).

4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input $(K_b, C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same types of oracles as $\mathcal{A}_1$. However, $\mathcal{A}_2$ may not make a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ query on the input (C^*, t^*) or a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query on the input (C^*, V_{C^*}). $\mathcal{A}_2$ terminates by outputing a guessing bit b'.

In this game the attacker's advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$.

Definition 18. *A TRE-PC KEM is IND-TR-KEM-CPA$_{OS}$ secure if it is IND-TR-KEM-CCA$_{OS}$ secure against attackers that make no decryption queries.*

5.5.2 Security against curious time server

In this subsection we define the security of a TRE-PC KEM against the curious time server. Specifically, we define the security under an adaptive chosen ciphertext attack (IND-TR-KEM-CCA$_{\mathrm{TS}}$ security) and the security under an adaptive chosen plaintext attack (IND-TR-KEM-CPA$_{\mathrm{TS}}$ security).

Definition 19. *A TRE-PC KEM is IND-TR-KEM-CCA$_{TS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ only has negligible advantage in the following game.*

1. Game setup: The challenger runs $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ to generate the time server's master key mk and the public parameters *param*. The challenger also runs $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input $(mk, pk_r, param)$. $\mathcal{A}_1$ has access to the following oracles.

   - $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ oracle, which, on receiving a query for (C, V'_C), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C, V'_C, sk_r)$. It should be noted that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key of C.
   - $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ oracle, which, on receiving a query for (C, t'), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, TS_{t'}, sk_r)$. It should be noted that t' need not be the legitimate release time for C.

   $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information *state*.

3. Challenge: The challenger generates the challenge as follows:

   (a) The challenger computes $(K_0, C^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$.
   (b) The challenger randomly selects $K_1 \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$.
   (c) The challenger randomly selects a bit $b \in \{0,1\}$, and returns (K_b, C^*, V_{C^*}).

4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input $(K_b, C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same types of oracles as $\mathcal{A}_1$. However, $\mathcal{A}_2$ may not make a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ query on the input (C^*, t^*) or a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query on the input (C^*, V_{C^*}). $\mathcal{A}_2$ terminates by outputing a guessing bit b'.

In this game the attacker's advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$.

Definition 20. *A TRE-PC KEM is IND-TR-KEM-CPA$_{TS}$ secure if it is IND-TR-KEM-CCA$_{TS}$ secure against attackers that make no decryption queries.*

5.5.3 Security against malicious receiver

The security against the malicious receiver, i.e. IND-TR-KEM-CPA$_{\mathrm{IS}}$ security, is defined as follows.

Definition 21. *A TRE-PC KEM is IND-TR-KEM-CPA$_{IS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ only has negligible advantage in the following game.*

1. Game setup: The challenger runs $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$. $\mathcal{A}_1$ has access to the $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ oracle, which, on receiving a query for time t, returns $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}(mk, t)$. $\mathcal{A}_1$ terminates by outputting a release time t^* which is larger than all the inputs to the $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ oracle and some state information *state*.

3. Challenge: The challenger generates the challenge as follows:

   (a) The challenger computes $(K_0, C^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$.

   (b) The challenger randomly selects $K_1 \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$.

   (c) The challenger randomly selects a bit b, and returns (K_b, C^*).

4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input $(K_b, C^*, state)$. $\mathcal{A}_2$ has access to the $\mathsf{TRE\text{-}PC\text{-}KEM.Ext_{TS}}$ oracle on the input $t < t^*$. $\mathcal{A}_2$ eventually terminates by outputting a guessing bit b'.

In this game the attacker's advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$.

6 Constructing TRE-PC using TRE-PC KEM and DEM

In this section we first propose a method to build TRE-PC schemes using TRE-PC KEM and DEM, and then discuss the security of the proposed TRE-PC scheme.

6.1 The Construction

The polynomial-time algorithms of the proposed TRE-PC scheme are defined as follows.

- Setup algorithm is the same as the $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ algorithm.
- $\mathsf{Gen_U}$ algorithm is the same as the $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ algorithm.
- $\mathsf{Ext_{TS}}$ algorithm is the same as $\mathsf{TRE\text{-}PC\text{-}KEM.Ext_{TS}}$ algorithm.
- Enc algorithm: Taking a message m, a release time t, and the receiver's public key pk_r as input, this algorithm returns a ciphertext $C = (C_1, C_2)$ and a pre-open key V_C, where

$$(K, C_1, V_C) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t, pk_r),\ C_2 = \mathsf{DEM.Enc}(m, K).$$

- $\mathsf{Dec_{RK}}$ algorithm: Taking a ciphertext $C = (C_1, C_2)$, a pre-open key V_C, and the private key sk_r as input, this algorithm first computes K, where

$$K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap_{RK}}(C_1, V_C, sk_r).$$

  If $K = \perp$, the algorithm returns $\perp$. Otherwise, it returns m, where

$$m = \mathsf{DEM.Dec}(C_2, K),$$

- $\mathsf{Dec}_{\mathsf{PK}}$ algorithm: Taking a ciphertext $C = (C_1, C_2)$, the timestamp TS_t, and the private key sk_r as input, this algorithm first computes K, where

$$K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C_1, TS_t, sk_r).$$

If $K = \perp$, the algorithm returns $\perp$. Otherwise, it returns m, where

$$m = \mathsf{DEM.Dec}(C_2, K),$$

6.2 Security results

It is straightforward to verify that if the TRE-PC KEM and the DEM are sound then the TRE-PC scheme is sound.

Theorem 8. *If the TRE-PC KEM is binding, then the TRE-PC scheme is binding.*

Proof. Suppose $\mathcal{A}$ is a binding attacker for the TRE-PC scheme. We construct a binding attacker $\mathcal{A}'$ for the TRE-PC KEM which makes use of $\mathcal{A}$ as a subroutine. The attacker $\mathcal{A}'$ is defined as follows:

1. $\mathcal{A}'$ receives the public key pk_r and the public parameters $param$ as input and runs $\mathcal{A}$ on the input $(pk_r, param)$.
   - If $\mathcal{A}$ makes an extraction oracle query for a time t, then $\mathcal{A}'$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{A}$.
   - If $\mathcal{A}$ queries the $\mathsf{Dec}_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}'$ queries its $\mathsf{KEM.Decap}_{\mathsf{RK}}$ oracle on (C_1, V_C). If it receives $\perp$, $\mathcal{A}'$ returns $\perp$. Otherwise, if it receives K, $\mathcal{A}'$ returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}$.
   - If $\mathcal{A}$ queries the $\mathsf{Dec}_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then $\mathcal{A}'$ queries its $\mathsf{KEM.Decap}_{\mathsf{PK}}$ oracle on (C_1, t). If it receives $\perp$, $\mathcal{A}'$ returns $\perp$. Otherwise, if it receives K, $\mathcal{A}'$ returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}$.

   $\mathcal{A}$ terminates by outputting (C^*, V_{C^*}, t^*), where $C^* = (C_1^*, C_2^*)$.
2. $\mathcal{A}'$ terminates by outputting (C_1^*, V_{C^*}, t^*).

It is easy to see that $\mathcal{A}'$ is a legitimate binding attacker for the TRE-PC KEM and provides perfect simulation for the oracles that $\mathcal{A}$ may query. Now, suppose that $\mathcal{A}$ returns values $(C_1^*, C_2^*, V_{C^*}, t^*)$ such that

$$\perp \neq \mathsf{Dec}_{RK}(C^*, V_{C^*}, sk_r) \neq \mathsf{Dec}_{PK}(C^*, TS_{t^*}, sk_r) \neq \perp \ .$$

Then we must have that

$$\mathsf{KEM.Decap}_{RK}(C_1^*, V_{C^*}, sk_r) \neq \perp \qquad \mathsf{KEM.Decap}_{PK}(C_1^*, TS_{t^*}, sk_r) \neq \perp$$
$$\mathsf{KEM.Decap}_{RK}(C_1^*, V_{C^*}, sk_r) \neq \mathsf{KEM.Decap}_{PK}(C_1^*, TS_{t^*}, sk_r)\,.$$

The latter conditions comes from the fact that if $\mathsf{KEM.Decap}_{RK}(C_1^*, V_{C^*}, sk_r) = \mathsf{KEM.Decap}_{PK}(C_1^*, TS_{t^*}, sk_r) = K$ then

$$\mathsf{Dec}_{RK}(C^*, V_{C^*}, sk_r) = \mathsf{DEM.Dec}(C_2^*, K) = \mathsf{Dec}_{PK}(C^*, TS_{t^*}, sk_r)\,.$$

So, if $\mathcal{A}$ breaks the TRE-PC scheme, then $\mathcal{A}'$ breaks the TRE-PC KEM. However, the probability that $\mathcal{A}'$ breaks the TRE-PC KEM is negligible, and so we may deduce that the probability that $\mathcal{A}$ breaks the TRE-PC scheme is negligible, and that the TRE-PC scheme is binding. □

Theorem 9. *If the TRE-PC KEM is IND-TR-KEM-CCA$_{OS}$ secure and the DEM is IND-CCA2 secure, then the TRE-PC scheme is IND-TR-CCA$_{OS}$ secure.*

Proof. We prove the theorem through a sequence of games. Firstly, the legitimate IND-TR-CCA$_{\mathrm{OS}}$ attack game is described as follows.

1. Game Setup: The challenger runs Setup to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).
2. Phase 1: The attacker executes $\mathcal{A}_1$ on the input $(pk_r, param)$. $\mathcal{A}_1$ has access to the following oracles:
   - $\mathsf{Ext_{TS}}$ oracle, which takes as input a release time t, returns $\mathsf{Ext_{TS}}(mk, t)$.
   - $\mathsf{Dec_{RK}}$ oracle, which takes as input a ciphertext C and a pre-open key V_C, returns $\mathsf{Dec_{RK}}(C, V_C, sk_r)$.
   - $\mathsf{Dec_{PK}}$ oracle, which takes as input a ciphertext C and a release time t, returns $\mathsf{Dec_{PK}}(C, TS_t, sk_r)$.

   $\mathcal{A}_1$ terminates by outputting two equal length message m_0 and m_1, a release time t^* and some state information $state$.
3. Challenge: The challenger returns $C^* = (C_1^*, C_2^*)$ and a pre-open key V_{C^*}, which are computed in two steps:
   (a) Compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,
   (b) Compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^*)$.
4. Phase 2: The attacker executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same types of oracles as $\mathcal{A}_1$. However, $\mathcal{A}_2$ may not query the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}) or the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*). $\mathcal{A}_2$ terminates by outputting a guessing bit b'.

Let $\mathsf{Game_0}$ denote this legitimate game and E_0 be the event that $b = b'$ at the end of the game.

Secondly, consider a new game $\mathsf{Game_1}$ which is essentially identical to $\mathsf{Game_0}$, except for the following changes.

1. In the Challenge phase, the challenger returns $C^* = (C_1^*, C_2^*)$ and a pre-open key V_{C^*}, which are computed as follows:
   (a) Compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,
   (b) Randomly select $K^\dagger \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$, compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^\dagger)$.
2. In Phase 2, on receiving a $\mathsf{Dec_{RK}}$ query on the input (C, V_C), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $V_C = V_{C^*}$, the challenger returns $\mathsf{DEM.Dec}(C_2, K^\dagger)$.
3. In Phase 2, on receiving a $\mathsf{Dec_{PK}}$ query on the input (C, t), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $t = t^*$, the challenger returns $\mathsf{DEM.Dec}(C_2, K^\dagger)$.

Let E_1 be the event that $b = b'$ at the end of $\mathsf{Game_1}$. The following lemma shows that $|Pr[E_0] - Pr[E_1]|$ is negligible.

Lemma 14. *$|Pr[E_0] - Pr[E_1]|$ is negligible if the TRE-PC KEM is IND-TR-KEM-CCA$_{OS}$ secure.*

Proof. We construct an IND-TR-KEM-CCA$_{\mathrm{OS}}$ attacker $\mathcal{A}' = (\mathcal{A}_1', \mathcal{A}_2')$ for the TRE-PC KEM, which makes use of an IND-TR-CCA$_{\mathrm{OS}}$ attacker $\mathcal{A}$ as a subroutine, has the advantage $\frac{1}{2}|Pr[E_0] - Pr[E_1]|$. Hence, we will be able to conclude that $|Pr[E_0] - Pr[E_1]|$ is negligible as the TRE-PC KEM is IND-TR-KEM-CCA$_{\mathrm{OS}}$ secure.

$\mathcal{A}'$ takes $(pk_r, param)$, which is generated by the challenger as input, and consists of two sub-algorithms: $\mathcal{A}_1'$ and $\mathcal{A}_2'$, where

1. $\mathcal{A}_1'$ executes $\mathcal{A}_1$ on the input $(pk_r, param)$ and answers $\mathcal{A}_1$'s oracle queries as follows.
   - If $\mathcal{A}_1$ makes an extraction oracle query for a time t, then $\mathcal{A}_1'$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{A}_1$.
   - If $\mathcal{A}_1$ queries the $\mathsf{Dec_{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}_1'$ queries its $\mathsf{TRE\text{-}PC\text{-}KEM.Decap_{RK}}$ oracle on (C_1, V_C). If it receives $\perp$, $\mathcal{A}_1'$ returns $\perp$. Otherwise, if it receives K, $\mathcal{A}_1'$ returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}_1$.
   - If $\mathcal{A}_1$ queries the $\mathsf{Dec_{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then $\mathcal{A}_1'$ queries its $\mathsf{TRE\text{-}PC\text{-}KEM.Decap_{PK}}$ oracle on (C_1, t). If it receives $\perp$, $\mathcal{A}_1'$ returns $\perp$. Otherwise, if it receives K, $\mathcal{A}_1'$ returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}_1$.

   Suppose $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$. $\mathcal{A}_1'$ terminates by outputting t^*, and the state information $state' = (state, m_0, m_1)$.
2. $\mathcal{A}_2'$ takes $(K_b, C_1^*, V_{C^*}, state')$ as input, where (K_b, C_1^*, V_{C^*}) is the challenge for the TRE-PC KEM. $\mathcal{A}_2'$ computes $C_2^* = \mathsf{DEM.Enc}(m_d, K_b)$ where d is randomly chosen from $\{0,1\}$, and sets $C^* = (C_1^*, C_2^*)$. $\mathcal{A}_2'$ executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}_1'$ except for the following two cases.

(a) If $\mathcal{A}_2$ queries the $\mathsf{Dec}_{\mathsf{PK}}$ oracle on the input (C, V_C), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $V_C = V_{C^*}$, $\mathcal{A}_2'$ returns $\mathsf{DEM.Dec}(C_2, K_b)$.

(b) If $\mathcal{A}_2$ queries the $\mathsf{Dec}_{\mathsf{PK}}$ oracle on the input (C, t), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $t = t^*$, $\mathcal{A}_2'$ returns $\mathsf{DEM.Dec}(C_2, K_b)$.

Suppose $\mathcal{A}_2$ eventually terminates by outputting a guessing bit d', then $\mathcal{A}_2'$ terminates by outputting a guessing bit b', where $b' = 1$ if $d' = d$ and $b' = 0$ if $d' \neq d$.

It is easy to see that $\mathcal{A}'$ is a legitimate IND-TR-KEM-CCA$_{\mathrm{OS}}$ attacker and its advantage can be denoted as $\frac{1}{2}|Pr[b' = 1|b = 0] - Pr[b' = 1|b = 1]|$. If $b = 0$, $\mathcal{A}'$ provides perfect simulation for the oracles that an IND-TR-CCA$_{\mathrm{OS}}$ attacker may query in Game_0, otherwise $\mathcal{A}'$ provides perfect simulation for the oracles that such an attacker may query in Game_1. Therefore, $Pr[b' = 1|b = 0] = Pr[E_0]$, $Pr[b' = 1|b = 1] = Pr[E_1]$, and the lemma gets proved. □

Finally, we show that $|Pr[E_1] - \frac{1}{2}|$ is negligible by the following lemma.

Lemma 15. *$|Pr[E_1] - \frac{1}{2}|$ is negligible if the DEM is IND-CCA2 secure.*

Proof. We construct an IND-CCA2 attacker $\mathcal{A}' = (\mathcal{A}_1', \mathcal{A}_2')$ for the DEM, which makes use of an IND-TR-CCA$_{\mathrm{OS}}$ attacker $\mathcal{A}$ as a subroutine, has the advantage $|Pr[E_1] - \frac{1}{2}|$. Hence, we will be able to conclude that $|Pr[E_1] - \frac{1}{2}|$ is negligible as the DEM is IND-CCA2 secure.

$\mathcal{A}'$ takes a security parameter 1^ℓ as input and consists of two sub-algorithms: $\mathcal{A}_1'$ and $\mathcal{A}_2'$, where

1. $\mathcal{A}_1'$ takes 1^ℓ as input, runs Setup to generate mk and $param$, and runs $\mathsf{Gen}_{\mathsf{U}}$ to generate the public/private key pair (pk_r, sk_r). $\mathcal{A}_1'$ executes $\mathcal{A}_1$ on the input $(pk_r, param)$ and answers $\mathcal{A}_1$'s oracle queries as follows.

   - If $\mathcal{A}_1$ makes an extraction oracle query for a time t, then $\mathcal{A}_1'$ computes and returns the timestamp TS_t to $\mathcal{A}_1$.
   - If $\mathcal{A}_1$ queries the $\mathsf{Dec}_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}_1'$ computes $K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C_1, V_C)$. If $K = \perp$, $\mathcal{A}_1'$ returns $\perp$; otherwise, returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}_1$.
   - If $\mathcal{A}_1$ queries the $\mathsf{Dec}_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then $\mathcal{A}_1'$ computes $K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C_1, t)$. If $K = \perp$, $\mathcal{A}_1'$ returns $\perp$; otherwise, returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}_1$.

   Suppose $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$. $\mathcal{A}_1'$ terminates by outputting m_0 and m_1, and the state information $state' = (state, t^*)$.

2. $\mathcal{A}_2'$ takes $(C_2^*, state')$ as input, where C_2^* is the challenge for the DEM. $\mathcal{A}_2'$ computes $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$ and sets $C^* = (C_1^*, C_2^*)$. $\mathcal{A}_2'$ executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}_1'$ except for the following two cases.

(a) If $\mathcal{A}_2$ queries the $\mathsf{Dec_{PK}}$ oracle on the input (C, V_C), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $V_C = V_{C^*}$, $\mathcal{A}_2'$ queries its $\mathsf{DEM.Dec}$ on the input of C_2 and sends the result to $\mathcal{A}_2$.

(b) If $\mathcal{A}_2$ queries the $\mathsf{Dec_{PK}}$ oracle on the input (C, t), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $t = t^*$, $\mathcal{A}_2'$ queries its $\mathsf{DEM.Dec}$ oracle on the input of C_2 and sends the result to $\mathcal{A}_2$.

Suppose $\mathcal{A}_2$ eventually terminates by outputting a bit b', then $\mathcal{A}_2'$ terminates by outputting b'.

It is easy to see that $\mathcal{A}'$ is a legitimate IND-CCA2 attacker and provides perfect simulation for the oracles that $\mathcal{A}$ may query. $\mathcal{A}'$'s advantage is equal to $\mathcal{A}$'s advantage in the game $\mathsf{Game_1}$, i.e. $|Pr[E_1] - \frac{1}{2}|$. Since the DEM is IND-CCA2 secure, $|Pr[E_1] - \frac{1}{2}|$ is negligible and the lemma gets proved. □

We have proved that both $|Pr[E_0] - Pr[E_1]|$ and $|Pr[E_1] - \frac{1}{2}|$ are negligible. As a result, $|Pr[E_0] - \frac{1}{2}|$ is also negligible and the theorem gets proved. □

Using exactly the same techniques as in proving the previous theorem, we can prove the the following theorem about the IND-TR-CCA$_{\mathrm{TS}}$ security.

Theorem 10. *If the TRE-PC KEM is IND-TR-KEM-CCA$_{TS}$ secure and the DEM is IND-CCA2 secure, then the TRE-PC scheme is IND-TR-CCA$_{TS}$ secure.*

Theorem 11. *If the TRE-PC KEM is IND-TR-KEM-CPA$_{OS}$ secure and the DEM is IND-CPA secure, then the TRE-PC scheme is IND-TR-CPA$_{OS}$ secure.*

Proof. We prove the theorem through a sequence of games. Firstly, the legitimate IND-TR-CPA$_{\mathrm{OS}}$ attack game is described as follows.

1. Game Setup: The challenger runs Setup to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker executes $\mathcal{A}_1$ on the input $(pk_r, param)$. $\mathcal{A}_1$ has access to the $\mathsf{Ext_{TS}}$ oracle, which takes as input a release time t, returns $\mathsf{Ext_{TS}}(mk, t)$. $\mathcal{A}_1$ terminates by outputting two equal length message m_0 and m_1, a release time t^* and some state information $state$.

3. Challenge: The challenger returns $C^* = (C_1^*, C_2^*)$ and a pre-open key V_{C^*}, which are computed in two steps:

   (a) Compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,

   (b) Compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^*)$.

4. Phase 2: The attacker executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$. $\mathcal{A}_2$ also has access to the $\mathsf{Ext_{TS}}$ oracle. $\mathcal{A}_2$ eventually terminates by outputting a guessing bit b'.

Let Game_0 denote this legitimate game and E_0 be the event that $b = b'$ at the end of the game.

Secondly, consider a new game Game_1 which is essentially identical to Game_0, except that in the Challenge phase, the challenger returns $C^* = (C_1^*, C_2^*)$ and a pre-open key V_{C^*}, which are computed in as follows:

1. Compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,
2. Randomly select $K^\dagger \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$, compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^\dagger)$.

Let E_1 be the event that $b = b'$ at the end of Game_1. The following lemma shows that $|Pr[E_0] - Pr[E_1]|$ is negligible.

Lemma 16. *$|Pr[E_0] - Pr[E_1]|$ is negligible if the TRE-PC KEM is IND-TR-KEM-CPA$_{OS}$ secure.*

Proof. We construct an IND-TR-KEM-CPA$_{\mathrm{OS}}$ attacker $\mathcal{A}' = (\mathcal{A}_1', \mathcal{A}_2')$ for the TRE-PC KEM, which makes use of an IND-TR-CPA$_{\mathrm{OS}}$ attacker $\mathcal{A}$ as a subroutine, that has the advantage $\frac{1}{2}|Pr[E_0] - Pr[E_1]|$. Hence, we will be able to conclude that $|Pr[E_0] - Pr[E_1]|$ is negligible as the TRE-PC KEM is IND-TR-KEM-CPA$_{\mathrm{OS}}$ secure.

$\mathcal{A}'$ takes a security $(pk_r, param)$ as input and consists of the following sub-algorithms:

1. $\mathcal{A}_1'$ executes $\mathcal{A}_1$ on the input $(pk_r, param)$ and answers $\mathcal{A}_1$'s extraction oracle query. On receiving an extraction query for a time t, $\mathcal{A}_1'$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{A}_1$. Suppose $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$. $\mathcal{A}_1'$ terminates by outputting t^*, and the state information $state' = (state, m_0, m_1)$.
2. $\mathcal{A}_2'$ takes $(K_b, C_1^*, V_{C^*}, state')$ as input, where (K_b, C_1^*, V_{C^*}) is the challenge for the TRE-PC KEM. $\mathcal{A}_2'$ computes $C_2^* = \mathsf{DEM.Enc}(m_d, K_b)$ where d is randomly chosen from $\{0,1\}$, and sets $C^* = (C_1^*, C_2^*)$. $\mathcal{A}_2'$ executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}_1'$. Suppose $\mathcal{A}_2$ eventually terminates by outputting a guessing bit d', then $\mathcal{A}_2'$ terminates by outputting a guessing bit b', where $b' = 1$ if $d' = d$ and $b' = 0$ if $d' \neq d$.

It is easy to see that $\mathcal{A}'$ is a legitimate IND-TR-KEM-CPA$_{\mathrm{OS}}$ attacker and its advantage can be denoted as $\frac{1}{2}|Pr[b' = 1|b = 0] - Pr[b' = 1|b = 1]|$. If $b = 0$, $\mathcal{A}'$ provides perfect simulation for the oracles that an IND-TR-CPA$_{\mathrm{OS}}$ attacker may query in Game_0, otherwise $\mathcal{A}'$ provides perfect simulation for the oracles that such an attacker may query in Game_1. Therefore, $Pr[b' = 1|b = 0] = Pr[E_0]$, $Pr[b' = 1|b = 1] = Pr[E_1]$, and the lemma gets proved. □

Finally, we show that $|Pr[E_1] - \frac{1}{2}|$ is negligible by the following lemma.

Lemma 17. *$|Pr[E_1] - \frac{1}{2}|$ is negligible if the DEM is IND-CPA secure.*

Proof. We construct an IND-CPA attacker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for the DEM, which makes use of an IND-TR-CPA$_{\mathrm{OS}}$ attacker $\mathcal{A}$ as a subroutine, has the advantage $|Pr[E_1] - \frac{1}{2}|$. Hence, we will be able to conclude that $|Pr[E_1] - \frac{1}{2}|$ is negligible as the DEM is IND-CPA secure.

$\mathcal{A}'$ takes a security parameter 1^ℓ as input and consists of the following sub-algorithms:

1. $\mathcal{A}'_1$ takes 1^ℓ as input, runs Setup to generate mk and $param$, and runs $\mathsf{Gen_U}$ to generate the public/private key pair (pk_r, sk_r). $\mathcal{A}'_1$ executes $\mathcal{A}_1$ on the input $(pk_r, param)$. If $\mathcal{A}_1$ makes an extraction oracle query for a time t, then $\mathcal{A}'_1$ computes and returns the timestamp TS_t to $\mathcal{A}_1$. Suppose $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$. $\mathcal{A}'_1$ terminates by outputting m_0 and m_1, and the state information $state' = (state, t^*)$.

2. $\mathcal{A}'_2$ takes $(C^*_2, state')$ as input, where C^*_2 is the challenge for the DEM. $\mathcal{A}'_2$ computes $(K^*, C^*_1, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$ and sets $C^* = (C^*_1, C^*_2)$. $\mathcal{A}'_2$ executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}'_1$. Suppose $\mathcal{A}_2$ eventually terminates by outputting a bit b', then $\mathcal{A}'_2$ terminates by outputting b'.

It is easy to see that $\mathcal{A}'$ is a legitimate IND-CPA attacker for the DEM and provides perfect simulation for the oracles that $\mathcal{A}$ may query. $\mathcal{A}'$'s advantage equals to $\mathcal{A}$'s advantage in the game $\mathsf{Game_1}$, i.e. $|Pr[E_1] - \frac{1}{2}|$. Since the DEM is IND-CPA secure, $|Pr[E_1] - \frac{1}{2}|$ is negligible and the lemma gets proved. □

We have proved that both $|Pr[E_0] - Pr[E_1]|$ and $|Pr[E_1] - \frac{1}{2}|$ are negligible. As a result, $|Pr[E_0] - \frac{1}{2}|$ is also negligible and the theorem gets proved. □

Using exactly the same techniques as in proving the previous theorem, we can prove the the following theorem about the IND-TR-CPA$_{\mathrm{TS}}$ security.

Theorem 12. *If the TRE-PC KEM is IND-TR-KEM-CPA$_{TS}$ secure and the DEM is IND-CPA secure, then the TRE-PC scheme is IND-TR-CPA$_{TS}$ secure.*

Theorem 13. *If the TRE-PC KEM is IND-TR-KEM-CPA$_{IS}$ secure and the DEM is IND-CPA secure, then the TRE-PC scheme is IND-TR-CPA$_{IS}$ secure.*

Proof. We prove the theorem through a sequence of games. Firstly, the legitimate IND-TR-CPA$_{\mathrm{IS}}$ attack game is described as follows.

1. Game Setup: The challenger runs Setup to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker executes $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$. $\mathcal{A}_1$ has access to the $\mathsf{Ext_{TS}}$ oracle, which takes as input a release time t, returns $\mathsf{Ext_{TS}}(mk, t)$. $\mathcal{A}_1$ terminates by outputting two equal length message m_0 and m_1, a release time t^*, which is larger than the the inputs to the $\mathsf{Ext_{TS}}$ oracle, and some state information $state$.

3. Challenge: The challenger returns $C^* = (C_1^*, C_2^*)$ which is computed as follows:

   (a) compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,

   (b) compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^*)$.

4. Phase 2: The attacker executes $\mathcal{A}_2$ on the input $(C^*, state)$. $\mathcal{A}_2$ has access to the $\mathsf{Ext_{TS}}$ oracle on any input $t < t^*$. $\mathcal{A}_2$ eventually terminates by outputting a guessing bit b'.

Let $\mathsf{Game_0}$ denote this legitimate game and E_0 be the event that $b = b'$ at the end of the game.

Secondly, consider a new game $\mathsf{Game_1}$ which is essentially identical to $\mathsf{Game_0}$, except that in the Challenge phase, the challenger returns $C^* = (C_1^*, C_2^*)$ which is computed as follows:

1. compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,

2. randomly select $K^\dagger \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$, compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^\dagger)$.

Let E_1 be the event that $b = b'$ at the end of $\mathsf{Game_1}$. The following lemma shows that $|Pr[E_0] - Pr[E_1]|$ is negligible.

Lemma 18. *$|Pr[E_0] - Pr[E_1]|$ is negligible if the TRE-PC KEM is IND-TR-KEM-CPA$_{IS}$ secure.*

Proof. We construct an IND-TR-KEM-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for the TRE-PC KEM, which makes use of an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}$ as a subroutine, has the advantage $\frac{1}{2}|Pr[E_0] - Pr[E_1]|$. Hence, we will be able to conclude that $|Pr[E_0] - Pr[E_1]|$ is negligible as the TRE-PC KEM is IND-TR-KEM-CPA$_{\mathrm{IS}}$ secure.

$\mathcal{A}'$ takes a security $(pk_r, sk_r, param)$ as input and consists of the following sub-algorithms:

1. $\mathcal{A}'_1$ executes $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$ and answers $\mathcal{A}_1$'s extraction oracle query. On receiving an extraction query for a time t, $\mathcal{A}'_1$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{A}_1$.

   Suppose $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$. $\mathcal{A}'_1$ terminates by outputting t^*, and the state information $state' = (state, m_0, m_1)$.

2. $\mathcal{A}'_2$ takes $(K_b, C_1^*, state')$ as input, where (K_b, C_1^*) is the challenge for the TRE-PC KEM. $\mathcal{A}'_2$ computes $C_2^* = \mathsf{DEM.Enc}(m_d, K_b)$ where d is randomly chosen from $\{0,1\}$, and sets $C^* = (C_1^*, C_2^*)$. $\mathcal{A}'_2$ executes $\mathcal{A}_2$ on the input $(C^*, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}'_1$. Suppose $\mathcal{A}_2$ eventually terminates by outputting a guessing bit d', then $\mathcal{A}'_2$ terminates by outputting a guessing bit b', where $b' = 1$ if $d' = d$ and $b' = 0$ if $d' \neq d$.

It is easy to see that $\mathcal{A}'$ is a legitimate IND-TR-KEM-CPA$_{\mathrm{IS}}$ attacker and its advantage can be denoted as $\frac{1}{2}|Pr[b'=1|b=0]-Pr[b'=1|b=1]|$. If $b=0$, $\mathcal{A}'$ provides perfect simulation for the oracles that an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}$ may query in Game_0, otherwise $\mathcal{A}'$ provides perfect simulation for the oracles that such an attacker may query in Game_1. Therefore, $Pr[b'=1|b=0]=Pr[E_0]$, $Pr[b'=1|b=1]=Pr[E_1]$, and the lemma gets proved. □

Finally, we show that $|Pr[E_1]-\frac{1}{2}|$ is negligible by the following lemma.

Lemma 19. *$|Pr[E_1]-\frac{1}{2}|$ is negligible if the DEM is IND-CPA secure.*

Proof. We construct an IND-CPA attacker $\mathcal{A}'=(\mathcal{A}'_1,\mathcal{A}'_2)$ for the DEM, which makes use of an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}$ as a subroutine, has the advantage $|Pr[E_1]-\frac{1}{2}|$. Hence, we will be able to conclude that $|Pr[E_1]-\frac{1}{2}|$ is negligible as the DEM is IND-CPA secure.

$\mathcal{A}'$ takes a security parameter 1^ℓ as input and consists of the following sub-algorithms:

1. $\mathcal{A}'_1$ takes 1^ℓ as input, runs Setup to generate mk and $param$, and runs $\mathsf{Gen_U}$ to generate the public/private key pair (pk_r, sk_r). $\mathcal{A}'_1$ executes $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$ and answers $\mathcal{A}_1$'s oracle queries in the same way as the challenger will in Game_1. Suppose $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$. $\mathcal{A}'_1$ terminates by outputting m_0 and m_1, and the state information $state'=(state,t^*)$.

2. $\mathcal{A}'_2$ takes $(C_2^*, state')$ as input, where C_2^* is the challenge for the DEM. $\mathcal{A}'_2$ computes $(K^*, C_1^*, V_{C^*})=\mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$ and sets $C^*=(C_1^*, C_2^*)$. $\mathcal{A}'_2$ executes $\mathcal{A}_2$ on the input $(C^*, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as the challenger will in Game_1. Suppose $\mathcal{A}_2$ eventually terminates by outputting a bit b', then $\mathcal{A}'_2$ terminates by outputting b'.

It is easy to see that $\mathcal{A}'$ is a legitimate IND-CPA attacker for the DEM and provides perfect simulation for the oracles that $\mathcal{A}$ may query. $\mathcal{A}'$'s advantage equals to $\mathcal{A}$'s advantage in the game Game_1, i.e. $|Pr[E_1]-\frac{1}{2}|$. Since the DEM is IND-CPA secure, $|Pr[E_1]-\frac{1}{2}|$ is negligible and the lemma gets proved. □

We have proved that both $|Pr[E_0]-Pr[E_1]|$ and $|Pr[E_1]-\frac{1}{2}|$ are negligible. As a result, $|Pr[E_0]-\frac{1}{2}|$ is also negligible and the theorem gets proved. □

7 Proposal of a TRE-PC KEM

7.1 The Description

The polynomial-time algorithms of the proposed TRE-PC-KEM are defined as follows.

- TRE-PC-KEM.Setup: This algorithm takes a security parameter 1^ℓ as input and generates the following parameters:
  - an additive group $\mathbb{G}_1$ of prime order q, a generator P of $\mathbb{G}_1$, and a multiplicative group $\mathbb{G}_2$ of the same order as $\mathbb{G}_1$,
  - a polynomial-time computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$,
  - three cryptographic hash functions $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1$, $\mathsf{H}_2 : \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \to \{0,1\}^\ell$, $\mathsf{H}_3 : \mathbb{G}_1 \times \mathbb{G}_2 \to \{0,1\}^{\mathsf{KeyLen}(\ell)}$,
  - a public/private key pair (S, s), where $S = sP$ and s is randomly chosen from $\mathbb{Z}_q$,

  where the master secret $mk = s$ and the public parameters $param = (\mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, S, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$.
- TRE-PC-KEM.$\mathsf{Ext}_{\mathsf{TS}}$: This algorithm takes the master secret mk and a time t as input and returns $TS_t = s\mathsf{H}_1(t)$.
- TRE-PC-KEM.Gen: This algorithm takes the security parameter 1^ℓ as input, randomly chooses sk_r from $\mathbb{Z}_q$, and generates a public/private key pair (pk_r, sk_r) where $pk_r = sk_r P$.
- TRE-PC-KEM.Encap: This algorithm takes a release time t and the receiver's public key pk_r as input, and returns (K, C, V_C), which are computed as follows:
  1. Randomly choose r and v from $\mathbb{Z}_q$, compute $Q_t = \mathsf{H}_1(t)$, $C_1 = rP$, $C_2 = vP$, $X_1 = r \cdot pk_r$, $X_2 = \hat{e}(vS, Q_t)$,
  2. Compute $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$, $K = \mathsf{H}_3(X_1, X_2)$,
  3. Set $V_C = vQ_t$ and $C = (C_1, C_2, C_3)$.
- TRE-PC-KEM.$\mathsf{Dec}_{\mathsf{RK}}$: This algorithm takes a ciphertext $C = (C_1, C_2, C_3)$, the pre-open key $V_C = vQ_t$, and the private key sk_r as input, and runs as follows:
  1. Compute $X_1 = sk_r C_1$ and $X_2 = \hat{e}(S, V_C)$, and check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds,
  2. If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return an error message $\perp$.
- TRE-PC-KEM.$\mathsf{Dec}_{\mathsf{PK}}$: This algorithm takes a ciphertext $C = (C_1, C_2, C_3)$, the timestamp TS_t, and the private key sk_r as input, and runs as follows:
  1. Compute $X_1 = sk_r C_1$ and $X_2 = \hat{e}(C_2, TS_t)$, and check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds,
  2. If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return an error message $\perp$.

7.2 Security results

The security of the proposed scheme is based on the Bilinear Diffie-Hellman (BDH) assumption which is formally described as follows: Given a security parameter 1^ℓ, there exists a polynomial-time algorithm which takes 1^ℓ as input and outputs an additive group $\mathbb{G}_1$ of prime order q, a generator P of $\mathbb{G}_1$, a multiplicative group $\mathbb{G}_2$ of the same order as $\mathbb{G}_1$, and a polynomial-time computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. On the input of $(\mathbb{G}_1, \mathbb{G}_2, P, q, \hat{e})$ and a BDH challenge $(\alpha P, \beta P, \gamma P)$ where α, β, γ are randomly chosen from $\mathbb{Z}_q$, any polynomial-time attacker can only compute $\hat{e}(P, P)^{\alpha\beta\gamma}$ with a negligible probability.

It is easy to see that the BDH assumption implies the Computational Diffie-Hellman (CDH) assumption, which is formally described as follows: Given a security parameter 1^ℓ, there exists a polynomial-time algorithm which takes 1^ℓ as input and outputs an additive group $\mathbb{G}$ of prime order q and a generator P of $\mathbb{G}$. On the input of $(\mathbb{G}, P, q)$ and a CDH challenge $(\alpha P, \beta P)$ where α, β are randomly chosen from $\mathbb{Z}_q$, any polynomial-time attacker can only compute $\alpha\beta P$ with a negligible probability. In our security analysis, we will use the following adapted CDH assumption. Given a security parameter 1^ℓ, there exists a polynomial-time algorithm which takes 1^ℓ as input and outputs an additive group $\mathbb{G}_1$ of prime order q, a generator P of $\mathbb{G}_1$, a multiplicative group $\mathbb{G}_2$ of the same order as $\mathbb{G}_1$, and a polynomial-time computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. On the input of $(\mathbb{G}_1, \mathbb{G}_2, P, q, \hat{e})$ and a CDH challenge $(\alpha P, \beta P)$ where α, β are randomly chosen from $\mathbb{Z}_q$, any polynomial-time attacker can only compute $\alpha\beta P$ with a negligible probability.

In the rest of this section, we prove that the proposed TRE-PC KEM is secure in the sense of binding, IND-TR-KEM-CCA$_{\mathrm{TS}}$, and IND-TR-KEM-CPA$_{\mathrm{IS}}$. Then from the security definitions, it is straightforward to verify that the proposed scheme is also secure in the sense of other security notions.

Theorem 14. *If H_2 is collision-resistant, then the TRE-PC KEM is binding.*

Proof. Without loss of generality, suppose that at the end of the legitimate binding attack game the attacker outputs (C^*, t^*, V_{C^*}), where $C^* = (C_1^*, C_2^*, C_3^*)$. Based on the definition of decapsulation algorithms, the attacker can only win the game if the following two requirements are satisfied:

1. $X_2' \neq X_2''$, where $X_2' = \hat{e}(S, V_{C^*})$ and $X_2'' = \hat{e}(C_2^*, TS_{t^*})$.
2. $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2')$ and $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2'')$, where $X_1^* = sk_r C_1^*$.

Therefore, the attacker wins the game implies that it finds a collision for H_2. Under the assumption that H_2 is collision-resistant, it is clear that the attacker can only win the game with a negligible probability. □

Theorem 15. *The TRE-PC KEM is IND-TR-KEM-CCA$_{TS}$ secure in the random oracle model under the CDH assumption.*

Proof. We prove the theorem through a sequence of games. Firstly, by modeling the hash functions as random oracles, the legitimate IND-TR-KEM-CCA$_{\mathrm{TS}}$ attack game (referred to as Game_0) in the random oracle model for the TRE-PC KEM is described as follows.

1. Game setup: The challenger first runs TRE-PC-KEM.Setup to generate $mk = s$ and $param = (\mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, S, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, and then runs TRE-PC-KEM.Gen to generate a public/private key pair (pk_r, sk_r). The challenger simulates the random oracle H_1 as follows: The challenger maintains a list of vectors, each of them contains a message, its hash value, and an element from $\mathbb{Z}_q$. After receiving a request message, the challenger first checks its list to see whether the hash value for the request message has been queried. If the check succeeds, the challenger returns the stored value, otherwise, the challenger returns yP, where y a random chosen from $\mathbb{Z}_q$. In the meantime, the challenger stores the new vector, containing the request message, yP, and y, to the existing list. H_2 and H_3 are simulated in a similar way.

2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input $(mk, pk_r, param)$. $\mathcal{A}_1$ has access to the decapsulation oracles:

   - TRE-PC-KEM.Decap$_{\mathsf{RK}}$ oracle on the input (C, V_C), where $C = (C_1, C_2, C_3)$. The challenger returns TRE-PC-KEM.Decap$_{\mathsf{RK}}(C, V_C, sk_r)$.
   - TRE-PC-KEM.Decap$_{\mathsf{PK}}$ oracle on the input (C, t), where $C = (C_1, C_2, C_3)$. Since the challenger knows s, it computes $TS_t = s\mathsf{H}_1(t)$ and returns TRE-PC-KEM.Decap$_{\mathsf{PK}}(C, TS_t, sk_r)$.

   At some point, $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information *state*.

3. Challenge: The challenger generates the challenge as follows:

   (a) Randomly choose r^* and v^* from $\mathbb{Z}_q$, K_1 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$,

   (b) Compute $Q_{t^*} = \mathsf{H}_1(t^*)$, $C_1^* = r^*P$, $C_2^* = v^*P$, $X_1^* = r^* \cdot pk_r$, $X_2^* = \hat{e}(v^*S, Q_{t^*})$, $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2^*)$, $K_0 = \mathsf{H}_3(X_1^*, X_2^*)$,

   (c) Return (K_b, C^*, V_{C^*}), where b is randomly chosen from $\{0,1\}$, $C^* = (C_1^*, C_2^*, C_3^*)$, and $V_{C^*} = v^*Q_{t^*}$.

4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input of $(K_b, C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the decapsulation oracles, but it may not make a TRE-PC-KEM.Decap$_{\mathsf{PK}}$ query on the input (C^*, t^*) or a TRE-PC-KEM.Decap$_{\mathsf{RK}}$ query on the input (C^*, V_{C^*}). $\mathcal{A}_2$ terminates by outputing a guessing bit b'.

Secondly, consider a new game Game_1 which is essentially identical to Game_0 except that the decapsulation queries are answered as follows.

- On receiving a TRE-PC-KEM.Decap$_{\mathsf{RK}}(C, V_C)$ query, where $C = (C_1, C_2, C_3)$, the challenger computes the response as follows.

1. Check whether $\mathcal{A}$ has queried the H_2 oracle on an input (z_1, z_2, z_3), where $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, $z_3 = \hat{e}(S, V_C)$.
2. If the check fails, return $\perp$; otherwise, set $X_1 = z_2, X_2 = z_3$ and continue to check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds.
3. If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return $\perp$.

- On receiving a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, t)$ query, where $C = (C_1, C_2, C_3)$, the challenger computes the response as follows.

1. Check whether whether $\mathcal{A}$ has queried the H_2 oracle on an input (z_1, z_2, z_3), where $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, $z_3 = \hat{e}(C_2, TS_t)$,
2. If the check fails, return $\perp$; otherwise, set $X_1 = z_2, X_2 = z_3$ and continue to check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds,
3. If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return $\perp$.

Let E_0 and E_1 denote the event $b = b'$ at the end of Game_0 and Game_1, respectively. The following lemma shows that $|Pr[E_0] - Pr[E_1]|$ is negligible.

Lemma 20. *$|Pr[E_0] - Pr[E_1]|$ is negligible in the random oracle model.*

Proof. Let F_1 be the event that, in step 2 of the attack game, the challenger answers any $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query on the input of (C, V_C) with a valid message, where $C = (C_1, C_2, C_3)$ and the attacker makes no H_2 query on the input (z_1, z_2, z_3) satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, $z_3 = \hat{e}(S, V_C)$. Given such a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query, it is straightforward to verify that the probability that $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ occurs with the probability $\frac{1}{2^\ell}$, where $C_1 = rP$, $X_1 = r \cdot pk_r$, and $X_2 = \hat{e}(S, V_C)$, because H_2 is modeled as a random oracle. As result, in the presence of a polynomial-time attacker, F_1 occurs with a negligible probability in Game_0.

Let F_2 be the event that, in step 2 of the attack game, the challenger answers any $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ query on the input of (C, t) with a valid message, where $C = (C_1, C_2, C_3)$ and the attacker makes no H_2 query on the input (z_1, z_2, z_3) satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, $z_3 = \hat{e}(C_2, TS_t)$. For similar reasons, $Pr[F_2]$ is negligible in Game_0.

Let F_3 be the event that, in step 4 of the attack game, the challenger answers any $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query on the input of (C, V_C) with a valid message, where $C = (C_1, C_2, C_3)$ and the attacker makes no H_2 query on the input (z_1, z_2, z_3) satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, $z_3 = \hat{e}(S, V_C)$. Recall that, for a valid query, either $C \neq C^*$ or $V_C \neq V_{C^*}$ should hold, therefore, at least one of the following inequations should hold: $C_1 \neq C_1^*$, $C_2 \neq C_2^*$, $C_3 \neq C_3^*$, and $V_C \neq V_{C^*}$. Given a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query on the input of (C, V_C) in Game_0, where the attacker makes no H_2 query on the input (z_1, z_2, z_3) satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, $z_3 = \hat{e}(S, V_C)$, based on the assumption that H_2 is regarded as a random oracle, the following facts are true:

1. If $C_1 \neq C_1^*$, then $X_1 \neq X_1^*$ and $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ occurs with the probability $\frac{1}{2^\ell}$.

2. If $C_2 \neq C_2^*$, then $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ occurs with the probability $\frac{1}{2^\ell}$.

3. If $V_C \neq V_{C^*}$, then $X_2 \neq X_2^*$ and $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ occurs with the probability $\frac{1}{2^\ell}$.

4. If $C_3 \neq C_3^*$, then $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ occurs at most with the probability $\frac{1}{2^\ell}$.

As result, in the presence of a polynomial-time attacker, F_3 occurs with a negligible probability in Game_0.

Let F_4 be the event that, in step 4 of the attack game, the challenger answers any $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ query on the input of (C, t) with a valid message, where $C = (C_1, C_2, C_3)$ and the attacker makes no H_2 query on the input (z_1, z_2, z_3) satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, $z_3 = \hat{e}(C_2, TS_t)$. For similar reasons, $Pr[F_4]$ is negligible in Game_0.

It is clear that Game_0 and Game_1 perform identically unless the event F_i, for some i $(1 \leq i \leq 4)$, occurs, therefore, $Pr[E_0|\neg(F_1 \vee F_2 \vee F_3 \vee F_4)] = Pr[E_1|\neg(F_1 \vee F_2 \vee F_3 \vee F_4)]$. Using the Difference Lemma introduced in [19], we immediately have $|Pr[E_0] - Pr[E_1]| \leq Pr[F_1 \vee F_2 \vee F_3 \vee F_4]$. Since $Pr[F_i$ $(1 \leq i \leq 4)$ are all negligible, $|Pr[E_0] - Pr[E_1]|$ is also negligible and the lemma gets proved. □

Thirdly, consider a new game Game_2 which is essentially identical to Game_1, except that the challenger randomly selects C_3^* from $\{0,1\}^\ell$, and K_0 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$ instead of computing $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2^*)$ and $K_0 = \mathsf{H}_3(X_1^*, X_2^*)$. Let E_2 be the event that $b = b'$ at the end of Game_2. The following lemma shows that $|Pr[E_1] - Pr[E_2]|$ is negligible.

Lemma 21. *$|Pr[E_1] - Pr[E_2]|$ is negligible in the random oracle model under the CDH assumption.*

Proof. It is clear that the games Game_2 and Game_1 perform identically unless H_2 is queried on the input of $(C_2^*, r^* \cdot pk_r, \hat{e}(v^*S, Q_{t^*}))$ or H_3 is queried on the input of $(r^* \cdot pk_r, \hat{e}(v^*S, Q_{t^*}))$. Let F_3 denote that either of the these events occurs. We now construct an algorithm $\mathcal{A}'$, which makes use of an IND-TR-KEM-CCA$_{\mathrm{TS}}$ attacker $\mathcal{A}$ as a subroutine, to solve the CDH problem with a non-negligible probability if $Pr[F_3]$ is non-negligible.

The attacker $\mathcal{A}'$ is implemented to play the same role as the challenger will play in Game_2:

1. $\mathcal{A}'$ receives the parameters $(\mathbb{G}_1, \mathbb{G}_2, P, q, \hat{e})$ and a CDH challenge $(\alpha P, \beta P)$, and generates $(\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, a public/private key pair (S, s) and sets $pk_r = \alpha P$. $\mathcal{A}'$ sets $param = (\mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, S, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, and simulates the random oracles in the same way as the challenger in Game_2.

2. $\mathcal{A}'$ runs $\mathcal{A}_1$ on the input of $(mk, pk_r, param)$ and answers $\mathcal{A}_1$'s decapsulation queries as follows.

- On receiving a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C, V_C)$ query, where $C = (C_1, C_2, C_3)$, the challenger computes the response as follows.
  (a) Check whether there is an input (z_1, z_2, z_3) to H_2 satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, $z_3 = \hat{e}(S, V_C)$,
  (b) If the check fails, return $\perp$; otherwise, set $X_1 = z_2, X_2 = z_3$ and continue to check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds,
  (c) If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return $\perp$.
- On receiving a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, t)$ query, where $C = (C_1, C_2, C_3)$, the challenger computes the response as follows.
  (a) Check whether there is an input (z_1, z_2, z_3) to H_2 satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, $z_3 = \hat{e}(C_2, TS_t)$,
  (b) If the check fails, return $\perp$; otherwise, set $X_1 = z_2, X_2 = z_3$ and continue to check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds,
  (c) If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return $\perp$.

3. When $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information *state*, $\mathcal{A}'$ computes the challenge as follows:
   (a) Randomly choose v^* from $\mathbb{Z}_q$, C_3^* from $\{0,1\}^\ell$, K_0 and K_1 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$,
   (b) Set $C_1^* = \beta P$ and compute $Q_{t^*} = \mathsf{H}_1(t^*)$, $C_2^* = v^* P$, $X_2^* = \hat{e}(v^* S, Q_{t^*})$,
   (c) Return (K_b, C^*, V_{C^*}), where b is randomly chosen from $\{0,1\}$, $C^* = (C_1^*, C_2^*, C_3^*)$, and $V_{C^*} = vQ_{t^*}$.

4. $\mathcal{A}'$ runs $\mathcal{A}_2$ on the input of $(K_b, C^*, V_{C^*}, state)$ and answers $\mathcal{A}_2$'s decapsulation queries in the same way as in step 2.

5. After $\mathcal{A}_2$ terminates, $\mathcal{A}'$ first randomly selects an input from the input set which is composed of the following two types of inputs: the inputs to H_2 in the form of $(C_2^*, ?, \hat{e}(v^* S, Q_{t^*}))$ and the inputs to H_3 in the form of $(?, \hat{e}(v^* S, Q_{t^*}))$, where ? can be any element from $\mathbb{G}_1$. If an input to H_2, say $(C_2^*, w_1', \hat{e}(v^* S, Q_{t^*}))$, is chosen, $\mathcal{A}'$ sets $\lambda = w_1'$, otherwise, if an input to H_3, say $(w_2', \hat{e}(v^* S, Q_{t^*}))$ is chosen then $\lambda = w_2'$. $\mathcal{A}'$ terminates by outputting λ.

It is easy to see that the algorithm $\mathcal{A}'$ faithfully plays the role that the challenger will play in Game_2. Suppose n_1 oracle queries has been made to H_2 and n_2 oracle queries has been made to H_3, where the queries are in the form specified in step 5 of $\mathcal{A}'$. Note the fact that the queries to H_3 are made either directly by $\mathcal{A}$ or by $\mathcal{A}'$ in simulating the decapsulation oracles, however, the queries to H_2 are all made by $\mathcal{A}$. Since $\mathcal{A}$ is a polynomial-time attacker, n_i $(1 \leq i \leq 2)$ are polynomials of ℓ. It is clear that $Pr[\lambda = \alpha\beta P] = \frac{1}{n_1 + n_2} Pr[F_3]$, so that $Pr[F_3]$ should be negligible based on the CDH assumption.

Since Game_2 and Game_1 perform identically unless the event F_3 occurs. Using the Difference Lemma [19], we have $|Pr[E_1]-Pr[E_2]| \leq Pr[F_3]$. As a result, $|Pr[E_1] - Pr[E_2]|$ is also negligible and the lemma gets proved. □

It is clear that $|Pr[E_2] - \frac{1}{2}| = 0$ in Game_2 because both K_0 and K_1 are randomly chosen. Therefore, we have proved that $|Pr[E_0] - Pr[E_1]|$, $|Pr[E_1] - Pr[E_2]|$, and $|Pr[E_2]-\frac{1}{2}|$ are negligible. As a result, $|Pr[E_0]-\frac{1}{2}|$ is also negligible and the theorem gets proved. □

Theorem 16. *The TRE-PC KEM scheme is IND-TR-KEM-CPA$_{IS}$ secure in the random oracle model under the BDH assumption.*

Proof. We prove the theorem through a sequence of games. Firstly, we describe the legitimate IND-TR-KEM-CPA$_{\mathrm{IS}}$ attack game in the random oracle model for the TRE-PC KEM.

1. Game setup: The challenger first runs $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ to generate $mk = s$ and $param = (\mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, S, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, and then runs $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ to generate a public/private key pair (pk_r, sk_r). The challenger simulates the random oracle H_1 as follows: The challenger maintains a list of vectors, each of them contains a message, its hash value, and an element from $\mathbb{Z}_q$. After receiving a request message, the challenger first checks its list to see whether the hash value for the request message has been queried. If the check succeeds, the challenger returns the stored value, otherwise, the challenger returns yP, where y a random chosen from $\mathbb{Z}_q$. In the meantime, the challenger stores the new vector, containing the request message, yP, and y, to the existing list. H_2 and H_3 are simulated in a similar way.

2. Phase 1: The attacker executes $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$. If $\mathcal{A}_1$ makes an extraction oracle query for a time t, the challenger returns the timestamp TS_t to $\mathcal{A}_1$. At some point, $\mathcal{A}_1$ terminates by outputting a release time t^*, which is larger than all the inputs to the $\mathsf{TRE\text{-}PC\text{-}KEM.Ext_{TS}}$ oracle, and some state information *state*.

3. Challenge: The challenger generates the challenge as follows:

   (a) Randomly choose r^* and v^* from $\mathbb{Z}_q$, K_1 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$,

   (b) Compute $Q_{t^*} = \mathsf{H}_1(t^*)$, $C_1^* = r^*P$, $C_2^* = v^*P$, $X_1^* = r^* \cdot pk_r$, $X_2^* = \hat{e}(v^*S, Q_{t^*})$, $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2^*)$, $K_0 = \mathsf{H}_3(X_1^*, X_2^*)$,

   (c) Return (K_b, C^*), where b is randomly chosen from $\{0,1\}$ and $C^* = (C_1^*, C_2^*, C_3^*)$.

4. Phase 2: The attacker executes $\mathcal{A}_2$ on the input $(K_b, C^*, state)$. If $\mathcal{A}_2$ makes an extraction oracle query for a time $t < t^*$, the challenger returns the timestamp TS_t to $\mathcal{A}_2$. $\mathcal{A}_2$ eventually terminates by outputting a bit b'.

Let Game_0 denote this legitimate game and E_0 be the event that $b = b'$ at the end of the game.

Secondly, consider a new game Game_1 which is essentially identical to Game_0, except that the challenger randomly selects C_3^* from $\{0,1\}^\ell$, and K_0 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$ instead of computing $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2^*)$ and $K_0 = \mathsf{H}_3(X_1^*, X_2^*)$. Let E_1 be the event that $b = b'$ at the end of Game_1. The following lemma shows that $|Pr[E_0] - Pr[E_1]|$ is negligible.

Lemma 22. *$|Pr[E_0] - Pr[E_1]|$ is negligible in the random oracle model under the BDH assumption.*

Proof. It is clear that the games Game_1 and Game_0 perform identically unless H_2 is queried on the input of $(C_2^*, r^* \cdot pk_r, \hat{e}(v^*S, Q_{t^*}))$ or H_3 is queried on the input of $(r^* \cdot pk_r, \hat{e}(v^*S, Q_{t^*}))$. Let F_1 denote that either of the these events occurs. We now construct an algorithm $\mathcal{A}'$, which makes use of an IND-TR-KEM-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}$ as a subroutine, to solve the BDH problem with a non-negligible probability if $Pr[F_1]$ is non-negligible.

Without loss of generality, we assume that the total number of H_1 queries $\mathcal{A}_1$ may make is bounded by n_1 ($n_1 \geq 1$)[1]. Note that these queries do not include those which are indirectly caused by the $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ queries. The attacker $\mathcal{A}'$ is implemented to play the same role as the challenger in Game_1 and defined as follows:

1. $\mathcal{A}'$ receives the parameters $(\mathbb{G}_1, \mathbb{G}_2, P, q, \hat{e})$ and a BDH challenge $(\alpha P, \beta P, \gamma P)$, and generates $(\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, a public/private key pair (S, s) and sets $S = \alpha P$. $\mathcal{A}'$ sets $param = (\mathbb{G}_1, \mathbb{G}_2, q, P, S, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, and simulates H_1, H_2, and H_3 in the same way as the challenger will do in Game_1. In addition, $\mathcal{A}'$ randomly selects $j \in \{1, 2, \cdots, n_1 + 1\}$.

2. $\mathcal{A}'$ runs $\mathcal{A}_1$ on the input of $(pk_r, sk_r, param)$. If $1 \leq j \leq n_1$, $\mathcal{A}'$ answers $\mathcal{A}_1$'s j-th query $\mathsf{H}_1(t')$ with γP, and stores $(t', \gamma P, null)$. If $\mathcal{A}_1$ makes a $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ query for a time t, $\mathcal{A}'$ first checks whether $t \neq t'$. If the check succeeds, $\mathcal{A}'$ computes and returns the timestamp TS_t to $\mathcal{A}_1$; otherwise, $\mathcal{A}'$ terminates as a failure. Note that if $t \neq t'$ then $\mathsf{H}_1(t) = yP$ for some y, where y is known to $\mathcal{A}'$. Therefore, $\mathcal{A}'$ can compute $TS_t = y\alpha P$ although α is unknown.

3. When $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information *state*, if $\mathcal{A}_1$ has not queried H_1 on t^*, then $\mathcal{A}'$ queries H_1 on t^* as the $(n_1 + 1)$-th query to this random oracle and sets $\mathsf{H}_1(t^*) = \gamma P$. If t^* was not the j-th query to the random oracle H_1, then $\mathcal{A}'$ terminates as a failure. Otherwise, $\mathcal{A}'$ computes the challenge as follows:

   (a) Randomly choose r^* from $\mathbb{Z}_q$, C_3^* from $\{0,1\}^\ell$, and K_0 and K_1 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$,

[1]For the simplicity of description, it is reasonable to require that $\mathcal{A}_1$ query H_1 only once with the same input.

(b) Compute $C_1^* = r^*P$ and set $C_2^* = \beta P$,

(c) Return (K_b, C^*), where b is randomly chosen from $\{0,1\}$ and $C^* = (C_1^*, C_2^*, C_3^*)$.

4. $\mathcal{A}'$ runs $\mathcal{A}_2$ on the input of $(K_b, C^*, state)$. If $\mathcal{A}_2$ makes any TRE-PC-KEM.$\mathsf{Ext}_{\mathsf{TS}}$ query on the input t, $\mathcal{A}'$ computes and returns the timestamp TS_t.

5. After $\mathcal{A}_2$ terminates, $\mathcal{A}'$ first randomly selects an input from the input set which is composed of the following two types of inputs: the inputs to H_2 in the form of $(C_2^*, r^* \cdot pk_r, ?)$ and the inputs to H_3 in the form of $(r^* \cdot pk_r, ?)$, where ? can be any element from $\mathbb{G}_2$. If an input to H_2, say $(C_2^*, r^* \cdot pk_r, w_1')$, is chosen, $\mathcal{A}'$ sets $\lambda = w_1'$, otherwise, if an input to H_3, say $(r^* \cdot pk_r, w_2')$ is chosen then $\lambda = w_2'$. $\mathcal{A}'$ terminates by outputting λ.

It is straightforward to verify that the probability that $\mathcal{A}'$ successfully ends is $\frac{1}{n_1+1}$, i.e. the probability that $\mathcal{A}'$ does not terminate in step 3 is $\frac{1}{n_1+1}$. If $\mathcal{A}'$ successfully ends, then it faithfully plays the role that the challenger will play in Game_1. Suppose n_2 oracle queries has been made to H_2 and n_3 oracle queries has been made to H_3, where the queries are in the form specified in step 5 of $\mathcal{A}'$. Note the fact that these queries are all made by $\mathcal{A}$. Since $\mathcal{A}$ is a polynomial-time attacker, n_i $(1 \leq i \leq 3)$ are polynomials of ℓ. Therefore, if $\mathcal{A}'$ successfully ends, the probability that $\lambda = \hat{e}(P,P)^{\alpha\beta\gamma}$ holds is $\frac{Pr[F_1]}{n_2+n_3}$, so that the probability that $\mathcal{A}'$ can compute $\hat{e}(P,P)^{\alpha\beta\gamma}$ is $\frac{Pr[F_1]}{(n_1+1)(n_2+n_3)}$. Based on the BDH assumption, $Pr[F_1]$ should be negligible. Recall that Game_1 and Game_0 perform identically unless the event F_1 occurs. Using the Difference Lemma [19], we have $|Pr[E_0] - Pr[E_1]| \leq Pr[F_1]$. As a result, $|Pr[E_0] - Pr[E_1]|$ is also negligible and the lemma gets proved. □

It is clear that $|Pr[E_1] - \frac{1}{2}| = 0$ in Game_1 because both K_0 and K_1 are randomly chosen. Therefore, we have proved that $|Pr[E_0] - Pr[E_1]|$ and $|Pr[E_1] - \frac{1}{2}|$ are negligible. Therefore, $|Pr[E_0] - \frac{1}{2}|$ is also negligible and the theorem gets proved. □

8 Conclusions

In this paper we have analysed a security model, i.e. the HYL model, for TRE-PC schemes proposed by Hwang, Yum, and Lee, and shown its defects. We proposed a new security model which avoids the defects possessed by the HYL model. Although the security definitions in Section 3.2 and 3.3 are only focused on either adaptive CCA/CPA attacks, it is straightforward to derive the definitions against non-adaptive CCA/CPA attacks. We also worked out the complete relations among the security notions defined in the proposed security model, introduced a new notion, i.e. TRE-PC KEM, and presented a hybrid model to construct TRE-PC schemes.

Acknowledgements

The second authour has been partially supported by a Thomas Holloway Studentship.

References

[1] M. Bellare and S. Goldwasser. Encapsulated key-escrow. Technical Report Tech. Report MIT/LCS/TR-688, MIT LCS, 1996.

[2] M. Bellare and S. Goldwasser. Verifiable partial key escrow. In *Proceedings of the 4th ACM conference on Computer and communications security*, pages 78–91. ACM Press, 1997.

[3] K. Bentahar, P. Farshim, J. Malone-Lee, and N.P. Smart. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive: Report 2005/058, 2005.

[4] T. E. Bjørstad and A. W. Dent. Building better signcryption schemes with tag-kems. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Proceedings of the 9th International Conference on Theory and Practice of Public-Key Cryptography, PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 491–507. Springer-Verlag, 2006.

[5] D. Boneh and M. Naor. Timed commitments. In M. Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, pages 236–254. Springer, 2000.

[6] J. Cathalo, B. Libert, and J.-J. Quisquater. Efficient and non-interactive timed-release encryption. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *Proceedings of the 7th International Conference on Information and Communications Security*, volume 3783 of *Lecture Notes in Computer Science*, pages 291–303. Springer-Verlag, 2005.

[7] A. C. F. Chan and I. F. Blake. Scalable, server-passive, user-anonymous timed release cryptography. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 504–513. IEEE Computer Society, 2005.

[8] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2004.

[9] G. D. Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 74–89. Springer-Verlag, 1999.

[10] A. W. Dent. A designer's guide to kems. In K. G. Paterson, editor, *Proceedings of the 9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151. Springer-Verlag, 2003.

[11] A. W. Dent. Hybrid signcryption schemes with insider security. In C. Boyd and J. M. G. Nieto, editors, *Proceedings of 10th Australasian Conference on Information Security and Privacy*, volume 3574 of *Lecture Notes in Computer Science*, pages 253–266. Springer-Verlag, 2005.

[12] A. W. Dent. Hybrid signcryption schemes with outsider security. In J. Zhou, J. Lopez, R. H. Deng, and F. Bao, editors, *Proceedings of the 8th International Information Security Conference, ISC 2005*, volume 3650 of *Lecture Notes in Computer Science*, pages 203–217. Springer-Verlag, 2005.

[13] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[14] Y. Hwang, D. Yum, and P. Lee. Timed-release encryption with pre-open capability and its application to certified e-mail system. In J. Zhou, J. Lopez, R. Deng, and F. Bao, editors, *Proceedings of the 8th International Information Security Conference (ISC 2005)*, volume 3650 of *Lecture Notes in Computer Science*, pages 344–358. Springer, 2005.

[15] T. C. May. *Time-release crypto*, 1993.

[16] R. C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[17] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report Tech. Report MIT/LCS/TR-684, MIT LCS, 1996.

[18] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In B. Preneel, editor, *Advances in Cryptology — Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 275–288. Springer-Verlag, 2000.

[19] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. http://shoup.net/papers/, 2006.