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

Alexander W. Dent[1] and Qiang Tang[1,2]

[1] Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK.
[2] Departement d'Informatique, École Normale Supérieure
45 Rue d'Ulm, 75230 Paris Cedex 05, France

Abstract. The concept of timed-released encryption with pre-open capability (TRE-PC) was introduced by Hwang, Yum and Lee. In a TRE-PC scheme, a message is encrypted in such a way that it can only be decrypted at a certain point in time or if the sender releases a piece of trapdoor information known as a pre-open key. This paper examines the security model for a TRE-PC scheme, demonstrates that a TRE-PC scheme can be constructed using a KEM–DEM approach, and provides an efficient example of a TRE-PC scheme.

1 Introduction

The concept of Timed-Release Encryption (TRE) is attributed to May [15]. In a TRE scheme, a message is encrypted in such a way that it can be decrypted by an authorised receiver only after a certain point in time. An unauthorised receiver should not be able to determine any information about the message from the ciphertext, and an authorised receiver should not be able to determine any information about the message before the stated release time. It is worth mentioning that some other timed primitives have been developed, for example, "price via processing" by Dwork and Naor [12], timed key escrow by Bellare and Goldwasser [1,2], and timed commitments by Boneh and Noar [5].

In the literature, there are two approaches used to construct TRE schemes. One approach is based on Merkle's time-lock puzzle technique [16] and involves encrypting the message in such a way that any computer attempting to decrypt the message will take at least a certain amount of time to solve the underlying computational problem [1,7,17]. 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. [6,9,17]). Generally, time-server-based schemes require interaction between the server and the users, and should prevent possible malicious behaviour of the time server. In this paper, we shall only be concerned with public-key TRE schemes that make use of time servers.

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] proposed such a scheme, which they term a *Timed-Release Encryption Scheme with Pre-Open Capability* (TRE-PC). In a TRE-PC scheme, a message is encrypted in such a way that it can only be decrypted at a certain point in time, or if the sender releases a piece of trapdoor information called a *pre-open key*. It should be infeasible for any user except for the intended receiver to determine any information about the message from the ciphertext, and the receiver should only be able to determine any information about the message after the release time or if they are given the 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 messege senders is not needed.

Rivest, Shamir, and Wagner gave a number of applications of Timed Released Encryption including electronic auctions, key escrow, chess moves, release of documents over time, payment schedules, press releases and etc. [17]. 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 most of these applications. Taking the electronic auction as an example, normally bidders in an auction seal their bid so that it can be opened after the bidding period is closed. However, if a bidder wishes to confirm their bid to the auctioneer at some point before the pre-defined open time, then they may come across some problems if a standard TRE scheme is adopted. Document escrow provides another useful example. Many legal systems require that classified governmental information is disclosed after a certain period of time. This can be achieved by using a TRE-PC scheme, through which the classified information can be encrypted by the public key of a special agent which is responsible for disclosing classified information. Note that no original classified information is required to be stored, and in the case that the information needs to be prematurely released, a pre-open key can be sent to the special agent which is able to decrypt the encrypted classified information.

Our contribution This paper makes three important contributions. First, we analyse the security model for TRE-PC schemes proposed by Hwang, Yum, and Lee [14] and show that it contains several deficiencies. We propose a new security model for a TRE-PC scheme. Second, we propose a new construction paradigm for a TRE-PC scheme based on the KEM-DEM approach of Cramer and Shoup [8,18]. We show that a TRE-PC scheme can be efficiently constructed from a TRE-PC KEM and a standard DEM. Lastly, we propose an efficient new TRE-PC KEM and prove its security in the random oracle model.

2 The Security Model for a TRE-PC Scheme

2.1 Notation

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers and $\{0,1\}^*$ the set of all bit strings. If $k \in \mathbb{N}$ then $\{0,1\}^k$ is the set of bit strings of length k and 1^k is the string of k ones. If $\mathcal{A}$ is a randomised algorithm, then $y \xleftarrow{\$} \mathcal{A}(x; O)$ denotes the assignment to y of the output of $\mathcal{A}$ when run on input x with fresh random coins and with access to oracle O; we write $y \leftarrow \mathcal{A}(x; O)$ if $\mathcal{A}$ is deterministic. If S is a finite set, then $x \xleftarrow{\$} S$ denotes the random generation of an element $x \in S$ using the uniform distribution. A function $\nu : \mathbb{N} \rightarrow [0, 1]$ is said to be *negligible* if for all $c \in \mathbb{N}$ there exists a $k_c \in \mathbb{N}$ such that $\nu(k) < k^{-c}$ for all $k > k_c$.

2.2 The HYL Security Model

In the paper that proposes the concept of timed-release encryption with pre-open capability, Hwang, Yum, and Lee [14] propose a security model against which the security of a TRE-PC scheme could be assessed. We refer to this model as the HYL model. A TRE-PC scheme proposed in the HYL model consists of six polynomial-time algorithms. A Setup algorithm is initially executed by a trusted time server. This algorithm outputs a series of system parameters and a master key for the time server. The time server uses this master key with the $\mathsf{Ext}_{\mathsf{TS}}$ algorithm to create a "timestamp" for a time t. A user generates their own encryption and decryption keys using the $\mathsf{Gen}_{\mathsf{PK}}$ algorithm. Encryption can then be performed using the Enc algorithm and a pre-open key generated using the $\mathsf{Gen}_{\mathsf{RK}}$ algorithm. These two algorithms must both take the same randomly generated secret value v as input if the pre-open key is going to help decrypt the ciphertext. Lastly, a ciphertext can be decrypted using the Dec algorithm, either using the appropriate timestamp or the pre-open key.

The HYL security model claims to consider two types of adversary: an outsider attacker (which could be "either a dishonest time server or an eavesdropper who tries to decrypt a legal receiver's ciphertext") and an inside attacker (who tries to decrypt a ciphertext before the release time without the pre-open key). Due to size constraints, we will not reproduce the HYL security models which can be found in the full version of the paper [11]. However, we suggest that the HYL model is incomplete and does not model all of the possible attacks that can be made against a TRE-PC scheme. In particular,

1. In the HYL model, the decryption process is described by one single algorithm, which works in two different modes depending on the input. We feel it is therefore more appropriate to formalise the decryption process as two independent algorithms.
2. In the HYL model, the means by which the secret value v used by the Enc and $\mathsf{Gen}_{\mathsf{RK}}$ algorithms is generated is never specified. We consider it more appropriate to remove the concept of a secret value, and have a single encryption algorithm that outputs both a ciphertext and the pre-open key for that ciphertext.

3. In the HYL model for an inside attacker, the attacker is able to obtain a timestamp for any time period except for the release time of the challenge ciphertext. In reality a receiver will only ever attempt to mount this attack before the release time of the challenge ciphertext. Hence, the HYL model is too strict. This is a problem as it is often advantageous if the timestamp for a given time period enables the receiver to decrypt all the messages that were encrypted for release at earlier times.
4. The HYL model does not give an outside attacker access to the pre-open key. However, it is realistic to assume that an outside attacker might be able to observe the pre-open key as it is being sent to the legitimate receiver.
5. The HYL model claims that an outside attacker captures the abilities of "either a dishonest time server or an eavesdropper who tries to decrypt a legal receiver's ciphertext". However, an outside attacker is not given access to the time server's master key and therefore does not model a dishonest time server.
6. A TRE-PC scheme allows the sender to release pre-open key which 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. This type of attack is not considered in the HYL model.

2.3 A new security model for TRE-PC schemes

We propose a new formulation and security model for a TRE-PC schemes. In our formulation, a TRE-PC scheme Π is given by six probabilistic, 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. Gen: 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. Ext: 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 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 ($\perp$).

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 and wish to break the confidentiality of a message.
- Curious time servers who knows the master key of the time server and wish to break the confidentiality of a message.
- 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.

This gives rise to four separate security models, shown in Fig. 1. All of these models mirror the standard definition for confidentiality in public-key encryption except for the binding model, which models the capability of an attacker to produce a ciphertext for which the two decryption algorithms return different messages. Each attacker may have access to one or more of the following oracles:

1. An Ext oracle that takes a time t as input and outputs the timestamp $TS_t = \mathsf{Ext}(t, mk)$.
2. A $\mathsf{Dec_{PK}}$ oracle that takes as input a ciphertext C and a time t, and outputs $\mathsf{Dec_{PK}}(C, TS_t, sk_r)$. Note that t need not be the "correct" release time for C.
3. A $\mathsf{Dec_{RK}}$ oracle that takes as input a ciphertext C and a pre-open key V, and outputs $\mathsf{Dec_{RK}}(C, V, sk_r)$. Note that V need not be the "correct" pre-open key for C.

For each of the IND games, a probabilistic, polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is deemed to have won if it outputs a value $b' = b$. $\mathcal{A}$'s advantage is defined to be $|Pr[b' = b] - 1/2|$. We may now formally define the security models for formalising the security against the above four types of adversaries.

Definition 1 (Outsider Security). *A TRE-PC scheme Π is said to be IND-TR-CCA$_{OS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*) or the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}) has negligible advantage.*

Definition 2 (Time Server Security). *A TRE-PC scheme Π is said to be IND-TR-CCA$_{TS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*) or the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}) has negligible advantage.*

Definition 3 (Insider Security). *A TRE-PC scheme Π is said to be IND-TR-CPA$_{IS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the Ext oracle on any time $t \geq t^*$ has negligible advantage.*

The use of the phrase 'CPA' in the definition of insider security may be misleading: since the attacker knows the user's secret key sk_r, the attacker does not gain any advantage from being given access to a $\mathsf{Dec_{RK}}$ oracle or a $\mathsf{Dec_{PK}}$ oracle for any time $t < t^*$. Hence, there is no point to proposing a IND-TR-CCA$_{\mathrm{IS}}$ security model. For the other two IND security definitions, we may propose analogous CPA definitions in the usual way.

Definition 4 (Binding). *A TRE-PC scheme Π is said to be binding if, for every polynomial-time attacker $\mathcal{A}$ that outputs a triple (C^*, t^*, V^*), we have that the probability that*

$$\perp \neq \mathsf{Dec_{PK}}(C^*, TS_{t^*}, sk_r) \neq \mathsf{Dec_{RK}}(C^*, V^*, sk_r) \neq \perp$$

is negligible.

It is worth stressing that we have adopted the notation "binding" which is a property of commitment schemes such as that in [5]. The binding property for TRE-PC schemes guarantees that, if the adversary has encrypted some message then it cannot release a pre-open key to force the receiver to decrypt a false message which is different from which was original sent. It is easy to see that this is an analog 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 later in our scheme). We further point out that if the receiver obtains $\perp$ in the decryption 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 of a TRE-PC scheme is also concerned with 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 has intended to send.

The relationship between these notions of security is given in Fig. 2. In this figure, "A $\longrightarrow$ B" means that if a scheme is secure in the sense of A then it is secure in the sense of B and "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. Given the relations in the figure, one can easily deduce the relation between any two security notions. Proofs of these relations can be found in the full version of the paper [11].

3 TRE-PC KEMs

The use of a symmetric encryption scheme as a subroutine of an asymmetric encryption schemes has long been known as a useful technique for improving the efficiency of asymmetric encryption. Cramer and Shoup [8,18] formalised one approach to producing such hybrid asymmetric encryption schemes. This

IND-TR-CCA$_{\mathbf{OS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(m_0, m_1, t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r; \mathsf{Ext}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Enc}(m_b, t^*, pk_r)$
6. $b' \xleftarrow{\$}$ $\mathcal{A}_2(C^*, V_{C^*}, state; \mathsf{Ext}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$

IND-TR-CCA$_{\mathbf{TS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(m_0, m_1, t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r, mk; \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Enc}(m_b, t^*, pk_r)$
6. $b' \xleftarrow{\$}$ $\mathcal{A}_2(C^*, V_{C^*}, state; \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$

IND-TR-CPA$_{\mathbf{IS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(m_0, m_1, t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r, sk_r; \mathsf{Ext})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Enc}(m_b, t^*, pk_r)$
6. $b' \xleftarrow{\$} \mathcal{A}_2(C^*, state; \mathsf{Ext})$

BINDING

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(C^*, t^*, V^*) \xleftarrow{\$}$ $\mathcal{A}(param, pk_r; \mathsf{Ext}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$

Fig. 1. Security models for a TRE-PC scheme

Fig. 2. Relations among the security notions

'KEM–DEM' approach has subsequently been applied to various other branches of asymmetric cryptography [3,4,10] and this section will explain how it can be applied to TRE-PC schemes.

A KEM-DEM scheme is composed of an asymmetric KEM and a symmetric DEM. The KEM random generates a symmetric key and an encapsulation (encryption) of that key. This symmetric key is then used by the DEM to encrypt a message. In this section, we first define a variant of KEM, namely, TRE-PC KEM, and then show that a secure TRE-PC scheme can be constructed from a secure TRE-PC KEM and a standard DEM.

3.1 Definitions of TRE-PC KEM

For the simplicity of description, the notation KEM refers to TRE-PC KEM in the following definition. A KEM consists of six probabilistic, polynomial-time algorithms:

- $\mathsf{KEM.Setup}$: This 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
- $\mathsf{KEM.Ext}$: This algorithm takes the master private key mk and a time t as input, and generates a timestamp TS_t.
- $\mathsf{KEM.Gen}$: This algorithm takes a security parameter 1^ℓ as input, and outputs a user's public/private key pair (pk_r, sk_r).
- $\mathsf{KEM.Encap}$: This 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.
- $\mathsf{KEM.Decap_{RK}}$: This 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 $\perp$.
- $\mathsf{KEM.Decap_{PK}}$: 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 $\perp$.

We assume that there exist a function $\mathsf{KeyLen}(\ell)$ such that the symmetric keys output by a particular TRE-PC KEM (with security parameter ℓ) are exactly $\mathsf{KeyLen}(\ell)$-bits long.

3.2 Security Definitions of TRE-PC KEM

Just as for a TRE-PC scheme, we actually define four separate security notions for a TRE-PC KEM, one for each of the different types of attacker. These security games are shown in Fig. 3. Once again, each attacker may have access to one or more of the following oracles:

1. An Ext oracle that takes a time t as input and outputs the timestamp $TS_t = \mathsf{Ext}(t, mk)$.

IND-TR-CCA$_{\mathbf{OS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r; \mathsf{Ext}, \mathsf{Decap}_{\mathsf{RK}}, \mathsf{Decap}_{\mathsf{PK}})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(K_0, C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Encap}(t^*, pk_r)$
6. $K_1 \xleftarrow{\$} \{0,1\}^{KeyLen(\ell)}$
7. $b' \xleftarrow{\$}$ $\mathcal{A}_2(K_b, C^*, V_{C^*}, state; \mathsf{Ext}, \mathsf{Dec}_{\mathsf{RK}}, \mathsf{Dec}_{\mathsf{PK}})$

IND-TR-CCA$_{\mathbf{TS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r, mk; \mathsf{Dec}_{\mathsf{RK}}, \mathsf{Dec}_{\mathsf{PK}})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(K_0, C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Encap}(t^*, pk_r)$
6. $K_1 \xleftarrow{\$} \{0,1\}^{KeyLen(\ell)}$
7. $b' \xleftarrow{\$}$ $\mathcal{A}_2(K_b, C^*, V_{C^*}, state; \mathsf{Dec}_{\mathsf{RK}}, \mathsf{Dec}_{\mathsf{PK}})$

IND-TR-CPA$_{\mathbf{IS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r, sk_r; \mathsf{Ext})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(K_0, C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Encap}(t^*, pk_r)$
6. $K_1 \xleftarrow{\$} \{0,1\}^{KeyLen(\ell)}$
7. $b' \xleftarrow{\$} \mathcal{A}_2(K_b, C^*, state; \mathsf{Ext})$

BINDING

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(C^*, t^*, V^*) \xleftarrow{\$}$ $\mathcal{A}(param, pk_r; \mathsf{Ext}, \mathsf{Dec}_{\mathsf{RK}}, \mathsf{Dec}_{\mathsf{PK}})$

Fig. 3. Security models for a TRE-PC KEM

2. A $\mathsf{Decap}_{\mathsf{PK}}$ oracle that takes as input an encapsulation C and a time t, and outputs either the encapsulated key K or an error message $\perp$.
3. A $\mathsf{Decap}_{\mathsf{RK}}$ oracle that takes as input an encapsulation C and a pre-open key V, and outputs either the encapsulated key K or an error message $\perp$.

For each of the IND games, a probabilistic, polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is deemed to have won if it outputs a value $b' = b$. $\mathcal{A}$'s advantage is defined to be $|Pr[b' = b] - 1/2|$.

The formal definitions for the security of a TRE-PC KEM mirror those of a full TRE-PC scheme:

Definition 5 (Outsider Security). *A TRE-PC KEM Π is said to be IND-TR-CCA$_{OS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the* $\mathsf{Decap}_{\mathsf{PK}}$ *oracle on the input (C^*, t^*) or the* $\mathsf{Decap}_{\mathsf{RK}}$ *oracle on the input (C^*, V_{C^*}) has negligible advantage.*

Definition 6 (Time Server Security). *A TRE-PC KEM Π is said to be IND-TR-CCA$_{TS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the* $\mathsf{Decap}_{\mathsf{PK}}$ *oracle on the input (C^*, t^*) or the* $\mathsf{Decap}_{\mathsf{RK}}$ *oracle on the input (C^*, V_{C^*}) has negligible advantage.*

Definition 7 (Insider Security). *A TRE-PC KEM Π is said to be IND-TR-CPA$_{IS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the* Ext *oracle on any time $t \geq t^*$ has negligible advantage.*

Definition 8 (Binding). *A TRE-PC KEM Π is said to be binding if, for every polynomial-time attacker $\mathcal{A}$ that outputs a triple (C^*, t^*, V^*), we have that the probability that*

$$\perp \neq \mathsf{Decap}_{\mathsf{PK}}(C^*, TS_{t^*}, sk_r) \neq \mathsf{Decap}_{\mathsf{RK}}(C^*, V^*, sk_r) \neq \perp$$

is negligible.

3.3 Construction of TRE-PC Schemes

As might be expected, we show that the combination of a secure TRE-PC KEM and a secure DEM is a secure TRE-PC scheme. We first recall the definition of a DEM [8,18]. A DEM consists of the following two polynomial-time algorithms:

- $\mathsf{DEM.Enc}$: A deterministic, polynomial-time encryption algorithm which, on the input a message m and a symmetric key K, outputs a ciphertext C.
- $\mathsf{DEM.Dec}$: A deterministic, polynomial-time decryption algorithm 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 TRE-PC KEM, i.e. $\{0,1\}^{\mathsf{KeyLen}(\ell)}$. We also assume that the TRE-PC KEM and DEM are sound in that the appropriate decapsulation/decryption algorithms 'undo' the effects of the encapsulation/encryption algorithms. We may now construct a TRE-PC scheme from a TRE-PC KEM and a DEM:

- The Setup, Ext, and Gen algorithms are given by the $\mathsf{KEM.Setup}$, $\mathsf{KEM.Ext}$, and $\mathsf{KEM.Gen}$ algorithms, respectively.
- The encryption algorithm $\mathsf{Enc}(m, t, pk_r)$ works in two steps. It first runs $(K, C_1, V_C) \xleftarrow{\$} \mathsf{KEM.Encap}(t, pk_r)$, and then computes $C_2 \leftarrow \mathsf{DEM.Enc}(m, K)$. The ciphertext is $C \leftarrow (C_1, C_2)$ and the pre-open key is V_C.
- The decryption algorithm $\mathsf{Dec}_{\mathsf{RK}}(C, V_C, sk_r)$ works in two steps. It first runs $K \leftarrow \mathsf{KEM.Decap}_{\mathsf{RK}}(C_1, V_C, sk_r)$, and then outputs the message $m \leftarrow \mathsf{DEM.Dec}(C_2, K)$. If $K = \perp$, this decryption outputs $\perp$.
- The decryption algorithm $\mathsf{Dec}_{\mathsf{PK}}(C, TS_t, sk_r)$ works in two steps. It first runs $K \leftarrow \mathsf{KEM.Decap}_{\mathsf{PK}}(C_1, TS_t, sk_r)$, and then outputs the message $m \leftarrow \mathsf{DEM.Dec}(C_2, K)$. If $K = \perp$, this decryption outputs $\perp$.

We also use the notion of one-time IND-CCA and IND-CPA security for a DEM that was proposed by Cramer and Shoup [8,18]. It is not difficult to see that we can now prove the following theorems about a TRE-PC constructed from a TRE-PC KEM and a DEM. The proofs for the IND security of the composition are similar to those of Cramer and Shoup [8,18] and can be found in the full version of the paper [11]. Note that TRE-PC KEM and and the DEM are trivially required to be sound.

Theorem 1. *Let a hybrid TRE-PC scheme be formed from a TRE-PC KEM and a DEM. If the TRE-PC KEM is IND-TR-CCA$_{OS}$ secure and the DEM is IND-CCA secure, then the TRE-PC scheme is IND-TR-CCA$_{OS}$ secure.*

Theorem 2. *Let a hybrid TRE-PC scheme be formed from a TRE-PC KEM and a DEM. If the TRE-PC KEM is IND-TR-CCA$_{TS}$ secure and the DEM is IND-CCA secure, then the TRE-PC scheme is IND-TR-CCA$_{TS}$ secure.*

Theorem 3. *Let a hybrid TRE-PC scheme be formed from a TRE-PC KEM and a DEM. If the TRE-PC KEM is IND-TR-CPA$_{IS}$ secure and the DEM is IND-CPA secure, then the TRE-PC scheme is IND-TR-CPA$_{IS}$ secure.*

Theorem 4. *Let a hybrid TRE-PC scheme be formed from a TRE-PC KEM and a DEM. If the TRE-PC KEM is binding, then the TRE-PC scheme is binding.*

4 An Efficient TRE-PC KEM

In this section, we propose a concrete instantiation of a TRE-PC KEM. The scheme we propose shares similarities with the scheme proposed by Hwang, Yum and Lee [14]; however, our scheme is substantially simpler and, when used with a suitable DEM, gives rise to a more efficient TRE-PC scheme.

4.1 The Description

Our scheme makes use of a bilinear map on a group. In other words, we assume the existence of an instance generating algorithm that, given a security parameter 1^ℓ, outputs a group description $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are additively written groups of prime order q, P is a generator of $\mathbb{G}_1$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a polynomial-time computable, non-degenerate, bilinear map. This is normally instantiated by a super-singular elliptic curve of small embedding degree; for more details the reader is referred to the paper of Galbraith, Paterson and Smart [13].

The algorithms of the TRE-PC KEM are defined as follows:

- KEM.Setup: This algorithm takes the security parameter 1^ℓ as input, generates a group structure $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ of the required security level and chooses three hash functions:

  $$\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1 \quad \mathsf{H}_2 : \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_T \to \{0,1\}^\ell \quad \mathsf{H}_3 : \mathbb{G}_1 \times \mathbb{G}_T \to \{0,1\}^{\mathsf{KeyLen}(\ell)}.$$

  The algorithm then chooses a random element $s \xleftarrow{\$} \mathbb{Z}_q$ and sets $S \leftarrow sP$. The public parameters are $param \leftarrow (\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, S)$; the master private key is $mk \leftarrow s$.
- KEM.Ext: This algorithm takes the master secret mk and a time t as input, and returns $TS_t \leftarrow s\mathsf{H}_1(t)$.

- $\mathsf{KEM.Gen}$: This algorithm randomly generates $x \xleftarrow{\$} \mathbb{Z}_q$, and outputs the public/private keys $sk_r \leftarrow x$ and $pk_r \leftarrow xP$.
- $\mathsf{KEM.Encap}$: 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 generate $r \xleftarrow{\$} \mathbb{Z}_q$ and $v \xleftarrow{\$} \mathbb{Z}_q$.
  2. Compute $Q_t \leftarrow \mathsf{H}_1(t)$, $X_1 \leftarrow r \cdot pk_r$, $X_2 \leftarrow \hat{e}(S, Q_t)^v$.
  3. Compute $C_1 \leftarrow rP$, $C_2 \leftarrow vP$, $C_3 \leftarrow \mathsf{H}_2(C_2, X_1, X_2)$.
  4. Compute $K \leftarrow \mathsf{H}_3(X_1, X_2)$, $V_C \leftarrow vQ_t$ and $C \leftarrow (C_1, C_2, C_3)$.
- $\mathsf{KEM.Dec_{RK}}$: This algorithm takes a ciphertext $C = (C_1, C_2, C_3)$, the pre-open key $V_C = vQ_t$, and the private key $sk_r = x$ as input, and runs as follows:
  1. Compute $X_1 \leftarrow xC_1$ and $X_2 \leftarrow \hat{e}(S, V_C)$.
  2. Check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$. If not, output $\perp$ and halt.
  3. Otherwise, return $K \leftarrow \mathsf{H}_3(X_1, X_2)$.
- $\mathsf{KEM.Dec_{PK}}$: This algorithm takes a ciphertext $C = (C_1, C_2, C_3)$, the timestamp TS_t, and the private key $sk_r = x$ as input, and runs as follows:
  1. Compute $X_1 \leftarrow xC_1$ and $X_2 = \hat{e}(C_2, TS_t)$.
  2. Check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$. If not, output $\perp$ and halt.
  3. Otherwise, return $K \leftarrow \mathsf{H}_3(X_1, X_2)$.

4.2 Security results

The security of our scheme is based on two principles: that it is infeasible for any attacker who does not know the private key $sk_r = x$ to compute the value $X_1 = xC_1^*$ and that it is infeasible for any attacker who does not know either the master private key, the pre-open key or the appropriate timestamp to compute the value $X_2 = \hat{e}(P, P)^{rvs}$ for the given value of C_2^*. We prove the security of our scheme in the random oracle model under the following assumptions:

Definition 9 (Computational Diffie-Hellman). *Given a group description $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ generated at a security level 1^ℓ and a pair of group elements $(\alpha P, \beta P)$, where $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$, the computational Diffie-Hellman (CDH) problem is to determine $\alpha\beta P$. The CDH assumption is that no probabilistic, polynomial-time algorithm can solve this problem with non-negligible probability.*

Definition 10 (Bilinear Diffie-Hellman). *Given a group description $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ generated at a security level 1^ℓ and a triple of group elements $(\alpha P, \beta P, \gamma P)$, where $\alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_q$, the Bilinear Diffie-Hellman (BDH) problem is to determine $\hat{e}(P, P)^{\alpha\beta\gamma}$. The BDH assumption is that no probabilistic, polynomial-time algorithm can solve this problem with non-negligible probability.*

These computational assumptions allow us to prove the follow theorems about the IND security of our scheme.

Theorem 5. *The TRE-PC KEM is IND-TR-CCA$_{TS}$ secure in the random oracle model under the CDH assumption.*

Proof. We construct an algorithm $\mathcal{B}$ which solves the CDH problem with non-negligible probability whenever $\mathcal{A}$ breaks the IND-TR-CCA$_{\mathrm{TS}}$ security of the TRE-PC KEM with non-negligible advantage. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an IND-TR-CCA$_{\mathrm{TS}}$ attacker with non-negligible advantage. $\mathcal{B}$ runs as follows:

1. Receive an instance of the group on which the CDH problem is to be solved $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ and a CDH challenge $(\alpha P, \beta P)$.
2. Game setup: Randomly select $s \xleftarrow{\$} \mathbb{Z}_q$ and set $S = sP$. The public parameters are $param \leftarrow (\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, S)$ and the master private key is $mk \leftarrow s$. Set the user's public key to be $pk_r \leftarrow \alpha P$.
3. Phase 1: $\mathcal{B}$ executes $\mathcal{A}_1$ on the input $(param, pk_r, mk)$. $\mathcal{A}_1$ has access to several oracles during its execution (we assume, without loss of generality, that $\mathcal{A}$ never queries the random oracles with the same value twice):
   - If $\mathcal{A}$ queries the random oracle $\mathsf{H_i}$ with a new input Z, then $\mathcal{B}$ random generates a value Y from the appropriate range, stores (Z, Y) in $\mathsf{H_i}$-list and returns Y.
   - If $\mathcal{A}$ queries the $\mathsf{KEM.Decap_{RK}}$ oracle on the input $C = (C_1, C_2, C_3)$ and V_C, then $\mathcal{B}$ runs as follows:
     (a) Check whether there exists an input $Z = (z_1, z_2, z_3)$ on the $\mathsf{H_2}$-list such that $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$ and $z_3 = \hat{e}(S, V_C)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
     (b) Check whether $C_3 = \mathsf{H_2}(z_1, z_2, z_3)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
     (c) If both checks succeed, then $\mathcal{B}$ returns $K \leftarrow \mathsf{H_3}(z_2, z_3)$ to $\mathcal{A}$.
   - If $\mathcal{A}$ queries the $\mathsf{KEM.Decap_{PK}}$ oracle on the input $C = (C_1, C_2, C_3)$ and t, then $\mathcal{B}$ runs as follows:
     (a) Compute $TS_t = s\mathsf{H_1}(t)$.
     (b) Check whether there exists an input $Z = (z_1, z_2, z_3)$ on the $\mathsf{H_2}$-list such that $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$ and $z_3 = \hat{e}(C_2, TS_t)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
     (c) Check whether $C_3 = \mathsf{H_2}(z_1, z_2, z_3)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
     (d) If both checks succeed, then $\mathcal{B}$ returns $K \leftarrow \mathsf{H_3}(z_2, z_3)$ to $\mathcal{A}$.

   $\mathcal{A}_1$ terminates by outputting a challenge release time t^* and some state information *state*.
4. Challenge: $\mathcal{B}$ sets $C_1^* \leftarrow \beta P$ (the CDH challenge element). $\mathcal{B}$ also randomly selects $v \xleftarrow{\$} \mathbb{Z}_q$, $C_3^* \xleftarrow{\$} \{0,1\}^{\ell}$, $K \xleftarrow{\$} \{0,1\}^{\mathsf{KeyLen}(\ell)}$ and sets $C_2^* \leftarrow vP$, $V_{C^*} \leftarrow v\mathsf{H_1}(t^*)$. . The challenge ciphertext is set as $C^* \leftarrow (C_1^*, C_2^*, C_3^*)$.
5. Phase 2: $\mathcal{B}$ executes $\mathcal{A}_2$ on the input $(K, C^*, V_{C^*}, state)$. During its execution, $\mathcal{A}_2$ may query several oracles, these oracle queries are answered in the same way as in Phase 1. $\mathcal{A}_2$ terminates by outputting a bit b'.
6. $\mathcal{B}$ random selects an input Z on either the $\mathsf{H_2}$-list or the $\mathsf{H_3}$-list in such a way that all inputs are equally likely to be chosen. If $Z = (z_1, z_2, z_3)$ is an input on the $\mathsf{H_2}$-list, then $\mathcal{B}$ outputs z_2. If $Z = (z_1, z_2)$ is an input on the $\mathsf{H_3}$-list, then $\mathcal{B}$ outputs z_1.

We analyse this algorithm and show two things. First, that the environment that $\mathcal{A}$ can only distinguish the simulated environment from a real attack environment with negligible probability (up until the point in which $\mathcal{A}$ makes a

critical query to a hash function). Second, if $\mathcal{A}$ succeeds in breaking the security of the TRE-PC KEM, then it must make a critical query with non-negligible probability and that such a critical query allows $\mathcal{B}$ to recover the solution to the CDH problem with non-negligible probability.

Suppose $\mathcal{A}$ makes at most q_i queries to the random oracles $\mathsf{H_i}$, q_{RK} queries to the $\mathsf{KEM.Decap}_{RK}$ oracle and q_{PK} queries to the $\mathsf{KEM.Decap}_{PK}$ oracle. We define a critical query to be either:

- a query (z_1, z_2, z_3) to the $\mathsf{H_2}$ oracle such that $z_1 = C_2^*$, $z_2 = \alpha\beta P$ and $z_3 = \hat{e}(S, V_{C^*})$, or
- a query to the (z_1, z_2) to the $\mathsf{H_3}$ oracle such that $z_1 = \alpha\beta P$ and $z_2 = \hat{e}(C_2^*, TS_{t^*})$.

Note that the simulation of the random oracles is perfect up until the point where a critical query is made. Again, up until the point where a critical query is made, the simulation of the $\mathsf{KEM.Decap}_{RK}$ algorithm is perfect unless $\mathcal{A}$ submits a query (C_1, C_2, C_3) and V_C to the decapsulation oracle such that $\mathcal{A}$ has not queried the $\mathsf{H_2}$ oracle on the input $z_1 = C_2$, $z_2 = r\alpha P$ and $z_2 = \hat{e}(S, V_C)$, where $C_1 = rP$, and yet $\mathsf{H_2}(z_1, z_2, z_3) = C_3$. It is clear that, since $\mathsf{H_2}$ is a random oracle, these conditions will hold with probability $1/2^\ell$, which is negligible. This argument can also be used to show that the simulation of $\mathsf{KEM.Decap}_{PK}$ is sufficient correct up until the point where a critical query is made.

Let n_2 and n_3 be the maximum number of possible entries on the $\mathsf{H_2}$- and $\mathsf{H_3}$-lists respectively. Note that

$$n_2 = q_2 \qquad \text{and} \qquad n_3 = q_3 + q_{RK} + q_{PK}$$

due to the way in which the decapsulation oracles are simulated. If a critical query is made, then $\mathcal{B}$ will output the solution to the CDH problem with probability at least $1/n_2 + n_3$. Let E be the event that a critical query is made and let E' be the event that the critical $\mathsf{H_3}$ query is made, and note that $Pr[E] \geq Pr[E']$. Since $\mathsf{H_3}$ is a random oracle, a standard argument shows that $Pr[E']$ is greater than or equal to $\mathcal{A}$'s advantage. Therefore, since $\mathcal{A}$ has non-negligible advantage, we must have that $\mathcal{B}$ has a non-negligible probability of output the solution to the CDH problem. □

Theorem 6. *The TRE-PC KEM scheme is IND-TR-CPA$_{IS}$ secure in the random oracle model under the BDH assumption.*

Proof. The proof of this theorem is similar to the proof of IND-TR-CCA$_{\mathrm{TS}}$ security, although slightly more complex. Again, we construct an algorithm $\mathcal{B}$ that solves the BDH problem with non-negligible probability whenever $\mathcal{A}$ breaks the IND-TR-CPA$_{\mathrm{IS}}$ security of the TRE-PC KEM with non-negligible advantage. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an IND-TR-CPA$_{\mathrm{IS}}$ attacker with non-negligible advantage.

Suppose that $\mathcal{A}$ makes at most q_i queries to the random oracles $\mathsf{H_i}$, and q_E queries to the $\mathsf{KEM.Ext}$ oracle. $\mathcal{B}$ will keep track of the queries made to the $\mathsf{H_i}$

oracle via a number of lists. In any execution of $\mathcal{B}$, we shall see that the $\mathsf{H_i}$-list has at most n_i elements on it, where

$$n_1 = q_1 + q_E + 1\,, \qquad n_2 = q_2 \qquad \text{and} \qquad n_3 = q_3\,.$$

$\mathcal{B}$ runs as follows:

1. Receive an instance of the group on which the BDH problem is to be solved $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ and a BDH challenge $(\alpha P, \beta P, \gamma P)$.
2. $\mathcal{B}$ randomly choose an integer $j \xleftarrow{\$} \{1, 2, \ldots, n_1\}$. This will define $\mathcal{B}$'s guess for the challenge release time.
3. Game setup: Set $S \leftarrow \alpha P$ and define the public parameters to be $param \leftarrow (\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e}, \mathsf{H_1}, \mathsf{H_2}, \mathsf{H_3}, S)$. Randomly select $x \xleftarrow{\$} \mathbb{Z}_q$, set $sk_r \leftarrow x$ and $pk_r \leftarrow xP$.
4. Phase 1: $\mathcal{B}$ executes $\mathcal{A}_1$ on the input $(param, pk_r, sk_r)$. $\mathcal{A}_1$ has access to several oracles during its execution (we assume, without loss of generality, that $\mathcal{A}$ never queries the random oracles with the same value twice):
   - If $\mathcal{A}$ (or the $\mathsf{KEM.Ext}$ oracle) queries the random oracle $\mathsf{H_1}$ with a new input t and this is not the j-th new query to the $\mathsf{H_1}$ oracle, then $\mathcal{B}$ random generates a value $y \xleftarrow{\$} \mathbb{Z}_q$, sets $Y \leftarrow yP$, stores (t, y, Y) in $\mathsf{H_1}$-list and returns Y.
   - If $\mathcal{A}$ (or the $\mathsf{KEM.Ext}$ oracle) queries the random oracle $\mathsf{H_1}$ with a new input t and this is the j-th new query to the $\mathsf{H_1}$ oracle, then $\mathcal{B}$ adds $(t, \perp, \gamma P)$ to the $\mathsf{H_1}$-list and returns γP to $\mathcal{A}$.
   - If $\mathcal{A}$ queries the random oracle $\mathsf{H_2}$ or $\mathsf{H_3}$ with a new input Z, then $\mathcal{B}$ random generates a value Y from the appropriate range, stores (Z, Y) in the appropriate $\mathsf{H_i}$-list and returns Y.
   - If $\mathcal{A}$ queries the $\mathsf{KEM.Ext}$ oracle on the time t, then compute $\mathsf{H_1}(t)$, extract the appropriate element y from the $\mathsf{H_1}$-list entry (t, y, Y), and returns yS to $\mathcal{A}$. If there exists no element y, i.e. if t was the j-th query to the $\mathsf{H_1}$ oracle, then $\mathcal{B}$ terminates its entire execution by outputting a random group element from $\mathbb{G}_T$.

   $\mathcal{A}_1$ terminates by outputting a challenge release time t^* and some state information *state*.
5. If the $\mathsf{H_1}$ oracle has not been queried on t^*, then $\mathcal{B}$ "queries" $\mathsf{H_1}$ on t^*.
6. If t^* is not the j-th query to the $\mathsf{H_1}$ oracle, then $\mathcal{B}$ terminates its entire execution by outputting a random group element from $\mathbb{G}_T$.
7. Challenge: $\mathcal{B}$ randomly chooses $r^* \xleftarrow{\$} \mathbb{Z}_q$ and sets $C_1^* \leftarrow r^* P$. $\mathcal{B}$ sets $C_2^* \leftarrow \beta P$ and randomly selects $C_3^* \xleftarrow{\$} \{0,1\}^\ell$ and $K \xleftarrow{\$} \{0,1\}^{\mathsf{KeyLen}(\ell)}$. The challenge ciphertext is defined to be $C^* = (C_1^*, C_2^*, C_3^*)$.
8. Phase 2: $\mathcal{B}$ executes $\mathcal{A}_2$ on the input $(K, C^*, state)$. During its execution, $\mathcal{A}_2$ may query several oracles, these oracle queries are answered in the same way as in Phase 1. $\mathcal{A}_2$ terminates by outputting a bit b'.
9. $\mathcal{B}$ random selects an input Z on either the $\mathsf{H_2}$-list or the $\mathsf{H_3}$-list in such a way that all inputs are equally likely to be chosen. If $Z = (z_1, z_2, z_3)$ is an input on the $\mathsf{H_2}$-list, then $\mathcal{B}$ outputs z_3. If $Z = (z_1, z_2)$ is an input on the $\mathsf{H_3}$-list, then $\mathcal{B}$ outputs z_2.

Again we show that the environment provided by $\mathcal{B}$ to $\mathcal{A}$ almost exactly simulates the attack environment in which $\mathcal{A}$ expects to run up until either the simulation terminates because t^* is not the j-th query to H_1 or a critical oracle query is made. We define a critical oracle query to be either:

- a query (z_1, z_2, z_3) to the H_2 oracle such that $z_1 = C_2^*$, $z_2 = r^*xP$ and $z_3 = \hat{e}(C_2^*, TS_{t^*}) = \hat{e}(P,P)^{\alpha\beta\gamma}$, or
- a query to the (z_1, z_2) to the H_3 oracle such that $z_1 = r^*xP$ and $z_2 = \hat{e}(C_2^*, TS_{t^*}) = \hat{e}(P,P)^{\alpha\beta\gamma}$.

We note that the simulation of the random oracles and the extraction oracle is perfect up until the point in which a critical oracle query is made. Furthermore, we note that the probability that we make an incorrect choice of j is $1/n_1$.

Let E be the event that a critical oracle query is made and let E' be the event that the critical H_3 oracle query is made. Note that $Pr[E] \geq Pr[E']$. Now, by a standard argument, since H_3 is a random oracle, $\mathcal{A}$'s advantage in breaking the IND-TR-CPA$_{\text{IS}}$ security of the TRE-PC KEM is less than or equal to $Pr[E']$. Furthermore, if E occurs, then $\mathcal{B}$ has at least a $1/n_2+n_3$ chance of outputting the correct solution to the CDH problem. This means that if $\mathcal{A}$ has a non-negligible advantage ϵ of breaking the IND-TR-CPA$_{\text{IS}}$ security of the TRE-PC KEM, then $\mathcal{B}$ has a non-negligible probability of at least $\epsilon/n_1(n_2 + n_3)$ of solving the CDH problem. □

The binding of the scheme can be proven directly in the random oracle model, or in the standard model under the following assumption.

Definition 11 (Collision Resistance). *A hash function* H *generated at security level* 1^ℓ *is collision resistance if the probability that any polynomial-time algorithm can find a pair of inputs* $x \neq y$ *such that* $\mathsf{H}(x) = \mathsf{H}(y)$ *is negligible as a function of the security parameter.*

Theorem 7. *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 $C^* = (C_1^*, C_2^*, C_3^*)$. Recalling the definitions of $\mathsf{KEM.Decap}_{\mathsf{RK}}$ and $\mathsf{KEM.Decap}_{\mathsf{PK}}$ from the previous section, the attacker wins the game only if $O_1 \neq O_2$, where

$$O_1 = \mathsf{H}_3(X_1, X_2'),\ O_2 = \mathsf{H}_3(X_1, X_2''),\ X_1 = sk_r C_1^*,\ X_2' = \hat{e}(S, V_{C^*}),$$

$$X_2'' = \hat{e}(C_2^*, TS_{t^*}),\ C_3^* = \mathsf{H}_2(C_2^*, X_1, X_2'), \text{ and } C_3^* = \mathsf{H}_2(C_2^*, X_1, X_2'').$$

If the attacker wins, then it is staightforward to verify that $X_2' \neq X_2''$; otherwise $O_1 = O_2$. Hence, if the attacker wins the game then this implies that the attacker can find a collision for H_2, where the two inputs are (C_2^*, X_1, X_2') and (C_2^*, X_1, X_2''). Under the assumption that H_2 is collision-resistant, it follows that the attacker can only win the game with a negligible probability. □

5 Conclusions

In this paper we have analysed the security 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. 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 work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

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. 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.
11. A. W. Dent and Q. Tang. Revisiting the security model for timed-release public-key encryption with pre-open capability. http://eprint.iacr.org/2006/306, 2006.
12. 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.
13. S.D. Galbraith, K.G. Paterson, and N.P. Smart. Pairings for cryptographers. http://eprint.iacr.org/2006/165, 2006.
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.