A Conditional Proxy Broadcast Re-Encryption Scheme Supporting Timed-Release

Kaitai Liang[1], Qiong Huang[2⋆], Roman Schlegel[1], Duncan S. Wong[1], and Chunming Tang[3]

[1] Department of Computer Science, City University of Hong Kong, China
kliang4@student.cityu.edu.hk, rschlegel@gmx.ch, duncan@cityu.edu.hk
[2] College of Informatics, South China Agricultural University, Guangzhou, China
csqhuang@alumni.cityu.edu.hk
[3] School of Mathematics and Information Science, Guangzhou University, China
ctang@gzhu.edu.cn

Abstract. To allow a delegator not only to delegate the keyword-controlled decryption rights of a broadcast encryption to a set of specified recipients, but also to control when the decryption rights will be delegated, in this paper, for the first time, we introduce a new notion called Timed-Release Conditional Proxy Broadcast Re-Encryption (TR-CPBRE). We also propose a concrete construction for TR-CPBRE which can be proven selective identity adaptive CCA secure under the (P, Q, f)-general decisional Diffie-Hellman exponent assumption, and chosen-time period chosen-ciphertext secure under the bilinear Diffie-Hellman assumption. When compared with the existing CPBRE and Timed-Release Proxy Re-Encryption (TR-PRE) schemes, our scheme achieves better efficiency, and enables the delegator to make a fine-grained delegation of decryption rights to multiple delegatees.

Keywords: timed-release encryption, unidirectional conditional proxy broadcast re-encryption, bilinear map.

1 Introduction

Introduced by May [23] and further elaborated by Rivest et al. [25], Timed-Release Encryption (TRE) is a kind of time-dependent encryption where even a designated recipient cannot decrypt a ciphertext before a semi-trusted time server releases a trapdoor associated with the release time of the encryptor's choice. It has been found to have many real-world applications, such as sealed-bid auctions [11] and electronic-voting. To date, there have been some papers [12,13,20,22,24] that have proposed different variants of TRE.

⋆ Q. Huang is supported by the National Natural Science Foundation of China (No. 61103232), the Research Fund for the Doctoral Program of Higher Education of China (No. 20114404120027), and the Foundation for Distinguished Young Talents in Higher Education of Guangdong, China (No. LYM11033).

The traditional TRE only supports single recipient that seems undesirable in practice as a message might be intended for several recipients simultaneously. In 2005, Cathalo et al. [10] proposed an efficient TRE scheme, in which an encryptor is allowed to encrypt a message to multi-recipient with the same release time. The scheme is applicable to many network applications. Suppose there is an international programming contest, such as ACM-ICPC[4] and Google Code Jam[5]. The participating teams that are located all over the world are managed by different universities. All teams will be granted access to the problem set in a specified time. To prevent some unfair issues incurred by the network congestion or delivery delay, the contest organizer might prefer to allow all universities and teams to receive the problem set before the beginning of the contest, but not to open the set prior to the pre-specified time.

The above problem can be solved by using [10] as follows. The organizer first specifies a release time RT for a semi-trusted time server. With knowledge of the public keys of all registered universities and teams, the organizer encrypts the problem set m (as well as RT) (e.g., $Enc(PK_A, m, RT)$, $Enc(PK_{G1}, m, RT)$), and further sends the resulting ciphertexts to each university. Upon receiving the ciphertexts from the organizer, the university keeps its own ciphertext locally, and then forwards the rest of ciphertexts to the corresponding teams (whose identities are recorded in the register list). When the release time has arrived, the time server will release a trapdoor τ (corresponding to RT) such that the universities and teams can access the problem set simultaneously (See Fig. 1).

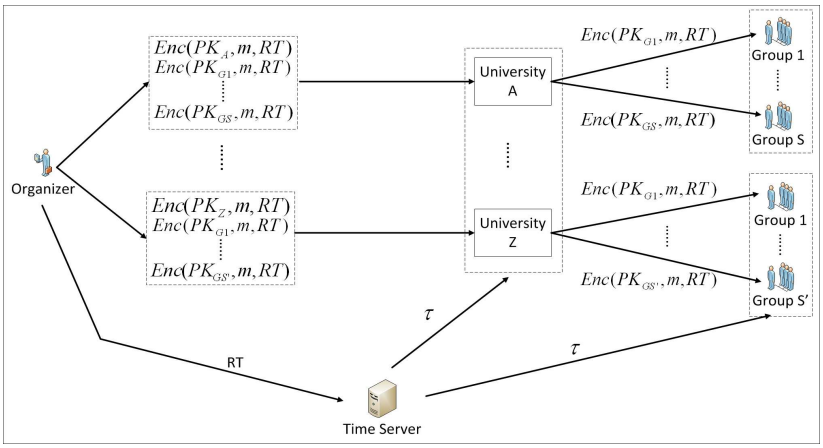

Fig. 1. Timed-Release Encryption for International Programming Contest

[4] http://icpc.baylor.edu/
[5] http://code.google.com/codejam/

The above solution, however, comes at a price that the organizer has to perform $n_1 \times (n_2 + 1)$ encryptions; meanwhile, $(n_2 + 1)$ ciphertexts are needed to be sent from the organizer to each university, where n_1 is the total number of university, and n_2 is the maximum number of team supervised by each university. This might be undesirable in practice due to the incurred linear communication complexity and computation cost.

To reduce the complexity, we might employ some existing cryptographic primitives in the above scenario. Intuitively, Broadcast Encryption (BE) that addresses the problem of confidentially broadcasting a message to a group of recipients might be one of candidates. Despite there exist some BE schemes (e.g., [5,6,18]) in the literature, it is unknown that whether these schemes can be extended to support timed-release property or not.

Proxy Re-Encryption (PRE) proposed by Blaze, Bleumer and Stauss [3], which increases the flexibility of data sharing, allows a semi-trusted proxy to transform a ciphertext intended for Alice into another ciphertext intended for Bob. The proxy, however, can learn nothing of the plaintext. PRE is applicable to many network applications, such as secure distributed files systems [1] and email forwarding encryption [3]. Since its introduction, many classic PRE schemes (e.g., [8,21,19]) have been proposed.

To employ PRE in the context of TRE, Emura et al. [17] proposed the first Timed-Release Proxy Re-Encryption (TR-PRE) that might be another candidate to solve the linear complexity problem. In TR-PRE, the proxy is allowed to re-encrypt a ciphertext with a release time under a public key to the one with the same release time under another public key by using a re-encryption key given by the delegator. Here we use $University\ A$ as an example. By uploading n_2 re-encryption keys (e.g., $rk_{A \to Group\ 1}, ..., rk_{A \to Group\ S}$) and its ciphertext $Enc(PK_A, m, RT)$ to the cloud (i.e. the proxy), $University\ A$ (i.e. the delegator) can request the proxy to re-encrypt the ciphertext to the ones intended for the teams under A's control (denoted the team set as $I_A = \{Group\ 1, ..., Group\ S\}$).

Despite TR-PRE allows the proxy to fulfill the re-encryption so as to relieve the workload of the organizer (who does not need to generate $(n_1 \times n_2)$ ciphertexts), the organizer still needs to generate n_1 encryptions for universities, and each university has to construct n_2 re-encryption keys. Moreover, without supporting any keyword (conditional) control on re-encryption, once given a re-encryption key (e.g., $rk_{A \to Group\ 1}$), the proxy can re-encrypt all ciphertexts of $University\ A$ to $Group\ 1$. This will incur the potential risk for access control.

Conditional Proxy Broadcast Re-Encryption (CPBRE), which was proposed by Chu et al. [14], can further reduce the cost incurred by TR-PRE. Specifically, CPBRE allows a delegator to delegate the decryption rights of a broadcast encryption to a set of delegatees, and to specify a condition to control the re-encryption power of the proxy. In CPBRE, only one (instead of n_2) re-encryption key is required to be generated by each university. Besides, the organizer only needs to generate one (instead of n_1) ciphertext for the university set (denoted as $I_U = \{University\ A, ..., University\ Z\}$). Thus CPBRE is an appropriate primitive for solving the linearly complexity problem. Nevertheless, the existing

CPBRE[6] cannot be trivially extended to support timed-release property due to the limitation of its proof technique. In the security proof given in [14], the challenger outputs a valid challenge ciphertext with the help of the challenger of an Hierarchical Identity-Coupling Broadcast Encryption (HICBE) [2]. To support timed-release, the challenge ciphertext has to be modified accordingly. This is out of the capability of the HICBE's challenger, that is, the challenger cannot output the corresponding challenge ciphertext. Therefore, a new CPBRE supporting timed-release property (i.e. Timed-Release Conditional Proxy Broadcast Re-Encryption (TR-CPBRE)) is desirable.

1.1 Our Contributions

In this paper, we formalize the definition and security models for TR-CPBRE. Specifically, a release time is required as an auxiliary input to the encryption and re-encryption key algorithms; meanwhile, this release time and its corresponding timed-release key are required in the input to the decryption algorithms. Note that a timed-release key is generated by a timed-release key generation algorithm that takes in the secret key of a semi-trusted time server and a given release time.

For security models, we consider two different aspects: one is to allow the adversary to get the timed-release key but not the secret key, that is, even if given the timed-release key, the adversary cannot decrypt a ciphertext without the appropriate secret key; the other is the inverse case, that is, even if given the secret key, the adversary cannot decrypt a corresponding ciphertext without the appropriate timed-release key. As of [14,17], we refer to the security of the former and the latter as *chosen ciphertext security* and *chosen-time-period and chosen-ciphertext security*, respectively.

Besides, we propose the first TR-CPBRE that is selective ID CCA (IND-sID-CCA) secure (under the (P, Q, f)-general decisional Diffie-Hellman exponent assumption), and is secure against chosen-time period chosen-ciphertext attacks (CTCA) (under the bilinear Diffie-Hellman assumption) in the random oracle model. In our scheme, the organizer is only needed to generate one ciphertext (with a release time RT and some condition c) for the university set I_U (i.e. $Enc(I_U, m, RT, c)$). To delegate the decryption rights of the broadcast encryption to its team set I_A, $University\ A$ first specifies to which group's broadcast encryption it would like to delegate (I_U), and next generates a re-encryption key from itself to I_A under c and RT (e.g., $rk_{A \to I_A|c,RT}$) for the proxy. The proxy then re-encrypts the ciphertext to the one ($Enc(I_A, m, RT, c)$) that can be only decrypted by the members of I_A (See Fig. 2). Note that the release time is still effective after the re-encryption. To the best of our knowledge, no TRE and PRE scheme (in general) capture timed-release and broadcast encryption properties simultaneously. TR-CPBRE is the first of its type.

Our scheme solves the limitations incurred by [17] in the sense that it only requires one re-encryption key and one re-encryption ciphertext rather than

[6] The only and available CPBRE due to Chu et al. is secure against Replayable Chosen Ciphertext Attacks (RCCA) [9].

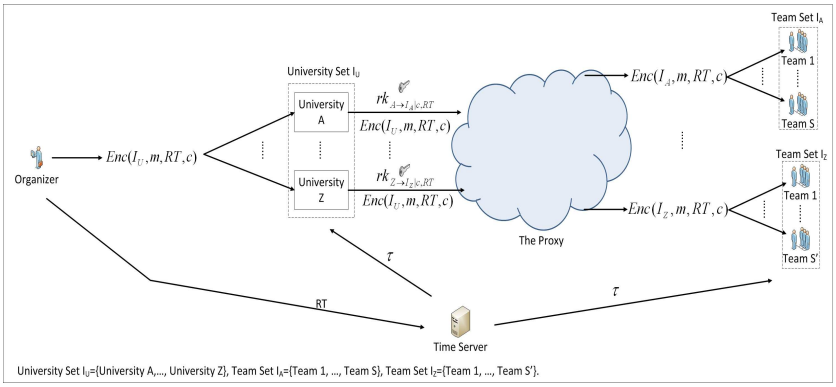

Fig. 2. Timed-Release Conditional Proxy Broadcast Re-Encryption

n_2 copies of them for each set of teams; meanwhile, it supports conditional control on re-encryption (i.e. conditional delegation). Thus our scheme enjoys improvement in communication compared to [17].

We argue that TR-CPBRE has many other real world applications, such as on-line learning systems (IXL[7]). For example, in an on-line learning system, the service provider can broadcast the learning materials in terms of different semesters with different release times to the universities which support on-line teaching, such that each university can accordingly open the classes in different semesters for its on-line learners.

We summarize the comparison of properties between our scheme, [14] and [17] in Table 1. While conditional delegation, broadcast re-encryption and timed-release property have been partially achieved by previous schemes, there is no CCA-secure proposal that achieves such properties simultaneously. However, this paper achieves the goal.

Table 1. Property Comparison

Schemes	Security	Selective Security	Conditional Delegation	Broadcast Re-Encryption	Timed-Release Property
CPBRE [14]	RCCA	✓	✓	✓	✗
TR-PRE [17]	RCCA	✗	✗	✗	✓
Our TR-CPBRE	**CCA**	✓	✓	✓	✓

[7] http://www.ixl.com/

2 Definition and Security Models

2.1 Definition of TR-CPBRE

Definition 1. *A (single-hop unidirectional) Timed-Release Conditional Proxy Broadcast Re-Encryption (TR-CPBRE) scheme consists of the following algorithms:*

1. $(param, msk, sk_{TS}) \leftarrow Setup(1^\lambda, n)$*: on input a security parameter* λ *and* n*, which indicates the maximum allowable number of receivers, output a public key* $param$*, a master secret key* msk*, a secret key* sk_{TS} *and a public key* TP *of a Time Server. Note that* TP *is regarded as one part of* $param$*.*
2. $sk_{ID} \leftarrow KeyGen(param, msk, ID)$*: on input* $param$*,* msk*, and an identity* $ID \in \{0,1\}^*$*, output a secret key* sk_{ID} *for identity* ID*.*
3. $\tau \leftarrow TS(sk_{TS}, RT)$*: on input* sk_{TS} *and a release time* $RT \in \{0,1\}^\lambda$*, output a timed-release key* τ*.*
4. $rk_{ID_i \to \overline{S}|RT,c} \leftarrow ReKeyGen(param, ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$*: on input* $param$*, an identity* ID_i *and the corresponding secret key* sk_{ID_i}*, two identity sets* S *and* $\overline{S}$*, a condition* $c \in \{0,1\}^*$ *and a release time* RT*, output a re-encryption key* $rk_{ID_i \to \overline{S}|RT,c}$*, where* $ID_i \in S$*.*
5. $C \leftarrow Enc(param, S, c, RT, m)$*: on input* $param$*, an identity set* S*,* c*,* RT *and a message* $m \in \{0,1\}^\lambda$*, output an original ciphertext* C*.*
6. $C_R \leftarrow ReEnc(param, rk_{ID_i \to \overline{S}|RT,c}, ID_i, S, \overline{S}, c, RT, C)$*: on input* $param$*, a re-encryption key* $rk_{ID_i \to \overline{S}|RT,c}$*, an identity* ID_i*, an identity set* S *such that* $ID_i \in S$*, an identity set* $\overline{S}$*,* c*,* RT *and* C*, output a re-encrypted ciphertext* C_R *or* $\perp$ *for failure.*
7. $m \leftarrow Dec(param, sk_{ID_i}, ID_i, S, c, RT, C, \tau)$*: on input* $param$*,* sk_{ID_i}*,* ID_i*,* S *such that* $ID_i \in S$*,* c*,* RT*, an original ciphertext* C *and* τ*, output a message* m *or* $\perp$ *for failure.*
8. $m \leftarrow Dec_R(param, sk_{\overline{ID}_{i'}}, ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R, \tau)$*: on input* $param$*, a delegatee's secret key* $sk_{\overline{ID}_{i'}}$*, a delegator's identity* ID_i*, a delegatee's identity* $\overline{ID}_{i'}$*, an identity set* S *such that* $ID_i \in S$*, an identity set* $\overline{S}$ *such that* $\overline{ID}_{i'} \in \overline{S}$*,* c*,* RT*, a re-encrypted ciphertext* C_R *and* τ*, output a message* m *or* $\perp$ *for failure.*

For simplicity, hereafter we omit *param* in the expression of the algorithms input.

Correctness: For any $\lambda, n \in \mathbb{N}$, any identity sets $S, \overline{S}$, any identities $ID_i, \overline{ID}_{i'} \in \{0,1\}^*$ such that $ID_i \in S$, $\overline{ID}_{i'} \in \overline{S}$, any condition $c \in \{0,1\}^*$, any release time $RT \in \{0,1\}^\lambda$ and any message $m \in \{0,1\}^\lambda$, if $(param, msk, sk_{TS}) \leftarrow Setup(1^\lambda, n)$, $\tau \leftarrow TS(sk_{TS}, RT)$, $sk_{ID} \leftarrow KeyGen(msk, ID)$, for all ID used in the system, $rk_{ID_i \to \overline{S}|RT,c} \leftarrow ReKeyGen(ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$, $C \leftarrow Enc(S, c, RT, m)$, and $C_R \leftarrow ReEnc(rk_{ID_i \to \overline{S}|RT,c}, ID_i, S, \overline{S}, c, RT, C)$, we have

$$Dec(sk_{ID_i}, ID_i, S, c, RT, C, \tau) = m;$$
$$Dec_R(sk_{\overline{ID}_{i'}}, ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R, \tau) = m.$$

2.2 Security Models

There are two main security requirements for TR-CPBRE: IND-sID-CCA security and CTCA security. Here we only give the security notions of original ciphertext. Note that the security notions of re-encrypted ciphertext can be defined in the same manner, we hence omit the details. We start with the formalization of IND-sID-CCA security. For capturing timed-release feature, in the model, we require that an adversary $\mathcal{A}$ is not able to win the game even if the time server's secret key sk_{TS} is known.

Definition 2. *A (single-hop unidirectional) TR-CPBRE scheme is IND-sID-CCA-secure at original ciphertext if no probabilistic polynomial time (PPT) adversary $\mathcal{A}$ can win the game below with non-negligible advantage. In the game, $\mathcal{C}$ is the game challenger, λ and n are the security parameter and the maximum allowable number of receivers, respectively.*

1. ***Initialization.*** *$\mathcal{A}$ outputs a challenge identity set $S^* = \{ID_1^*, ..., ID_s^*\}$, where $s \leq n$.*
2. ***Setup.*** *$\mathcal{C}$ runs $Setup(1^\lambda, n)$ and sends $param, sk_{TS}$ to $\mathcal{A}$.*
3. ***Phase 1.*** *$\mathcal{A}$ is given access to the following oracles.*
   (a) *$\mathcal{O}_{sk}(ID)$: on input an identity ID, output $sk_{ID} \leftarrow KeyGen(msk, ID)$. If $Extract(ID')$ is queried, we say that ID' is corrupted. One restriction is that $\mathcal{A}$ cannot query $Extract(ID)$ for any $ID \in S^*$.*
   (b) *$\mathcal{O}_{rk}(ID_i, S, \overline{S}, c, RT)$: on input an identity ID_i, two sets S and $\overline{S}$, a condition c and a release time RT, output $rk_{ID_i \to \overline{S}|RT,c} \leftarrow ReKeyGen(ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$, where $sk_{ID_i} \leftarrow KeyGen(msk, ID_i)$, $ID_i \in S$.*
   (c) *$\mathcal{O}_{re}(ID_i, S, \overline{S}, c, RT, C)$: on input an identity ID_i, two sets S and $\overline{S}$, a condition c, a release time RT, and an original ciphertext C, output $C_R \leftarrow ReEnc(rk_{ID_i \to \overline{S}|RT,c}, ID_i, S, \overline{S}, c, RT, C)$, where $rk_{ID_i \to \overline{S}|RT,c} \leftarrow ReKeyGen\ (ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$, $sk_{ID_i} \leftarrow KeyGen(msk, ID_i)$, $ID_i \in S$.*
   (d) *$\mathcal{O}_{dec}(ID_i, S, c, RT, C)$: on input an identity ID_i, an identity set S, a condition c, a release time RT, and an original ciphertext C, output $m \leftarrow Dec(sk_{ID_i}, ID_i, S, c, RT, C, \tau)$, where $sk_{ID_i} \leftarrow KeyGen(msk, ID_i)$, $\tau \leftarrow TS(sk_{TS}, RT)$, $ID_i \in S$.*
   (e) *$\mathcal{O}_{decr}(ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R)$: on input two identities ID_i and $\overline{ID}_{i'}$, two identity sets S and $\overline{S}$, a condition c, a release time RT, and a re-encrypted ciphertext C_R, output $m \leftarrow Dec_R(sk_{\overline{ID}_{i'}}, ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R, \tau)$, where $sk_{\overline{ID}_{i'}} \leftarrow KeyGen(msk, \overline{ID}_{i'})$, $\tau \leftarrow TS(sk_{TS}, RT)$, $\overline{ID}_{i'} \in \overline{S}$, $ID_i \in S$.*
4. ***Challenge.*** *$\mathcal{A}$ outputs two equal length messages m_0, m_1, a challenge condition c^* and a challenge release time RT^* to $\mathcal{C}$. If the queries $\mathcal{O}_{rk}(ID_i, S^*, \overline{S}, c^*, RT^*)$ and $\mathcal{O}_{sk}(\overline{ID}_{i'})$ are never made, $\mathcal{C}$ returns $C^* = Enc(S^*, c^*, RT^*, m_b)$ to $\mathcal{A}$, where $b \in_R \{0, 1\}$, $ID_i \in S^*$ and $\overline{ID}_{i'} \in \overline{S}$.*
5. ***Phase 2.*** *$\mathcal{A}$ continues making queries as in Phase 1 except the following:*
   (a) *$\mathcal{O}_{sk}(ID)$ for any $ID \in S^*$;*

(b) $\mathcal{O}_{rk}(ID_i, S^*, \overline{S}, c^*, RT^*)$ *and* $\mathcal{O}_{sk}(\overline{ID}_{i'})$ *for any* $ID_i \in S^*$ *and* $\overline{ID}_{i'} \in \overline{S}$;
(c) $\mathcal{O}_{dec}(ID_i, S^*, c^*, RT^*, C^*)$ *for any* $ID_i \in S^*$;
(d) $\mathcal{O}_{re}(ID_i, S^*, \overline{S}, c^*, RT^*, C^*)$ *and* $\mathcal{O}_{sk}(\overline{ID}_{i'})$ *for any* $ID_i \in S^*$, $\overline{ID}_{i'} \in \overline{S}$;
(e) $\mathcal{O}_{decr}(ID_i, \overline{ID}_{i'}, S^*, \overline{S}, c^*, RT^*, C_R)$ *for any* C_R, $ID_i \in S^*$, $\overline{ID}_{i'} \in \overline{S}$, *where* $(\overline{S}, c^*, RT^*, C_R)$ *is a derivative of* (S^*, c^*, RT^*, C^*). *As of [8], the derivative of* (S^*, c^*, RT^*, C^*) *is defined as follows.*
   i. (S^*, c^*, RT^*, C^*) *is a derivative of itself.*
   ii. *If* $\mathcal{A}$ *has issued a re-encryption key query on* $(ID_i, S^*, \overline{S}, c^*, RT^*)$ *to obtain* $rk_{ID_i \to \overline{S}|RT^*, c^*}$, *and* $C_R \leftarrow ReEnc(rk_{ID_i \to \overline{S}|RT^*, c^*}, ID_i, S^*, \overline{S}, c^*, RT^*, C^*)$, *then* $(\overline{S}, c^*, RT^*, C_R)$ *is a derivative of* (S^*, c^*, RT^*, C^*), *where* $ID_i \in S^*$.
   iii. *If* $\mathcal{A}$ *has issued a re-encryption query on* $(ID_i, S^*, \overline{S}, c^*, RT^*, C^*)$ *and obtained* C_R, *then* $(\overline{S}, c^*, RT^*, C_R)$ *is a derivative of* (S^*, c^*, RT^*, C^*).

6. ***Guess.*** $\mathcal{A}$ *outputs a guess bit* $b' \in \{0, 1\}$. *If* $b' = b$, $\mathcal{A}$ *wins.*

The advantage of $\mathcal{A}$ *is* $\epsilon = Adv^{IND-sID-CCA-Or}_{TR-CPBRE,\mathcal{A}}(1^\lambda, n) = |Pr[b' = b] - \frac{1}{2}|$.

Remark. If $\mathcal{A}$ is limited to output c^* and RT^* before the setup phase, then a static model can be captured. In such a model, $\mathcal{A}$ should follow the restriction, which is defined in the challenge phased, in Phase 1.

We now proceed to the IND-CTCA security.

Definition 3. *A (single-hop unidirectional) TR-CPBRE scheme is IND-CTCA-secure at original ciphertext if the advantage* $Adv^{IND-CTCA-Or}_{TR-CPBRE,\mathcal{A}}(1^\lambda, n)$ *is negligible for any PPT adversary* $\mathcal{A}$ *in the following experiment. Set* $O = \{\mathcal{O}_{sk}, \mathcal{O}_{rk}, \mathcal{O}_{ts}, \mathcal{O}_{re}, \mathcal{O}_{dec}, \mathcal{O}_{decr}\}$.

$$Adv^{IND-CTCA-Or}_{TR-CPBRE,\mathcal{A}}(1^\lambda, n) = |Pr[b' = b : (param, msk, sk_{TS}) \leftarrow Setup(1^\lambda, n); (m_0, m_1, S^*, RT^*, c^*, State) \leftarrow A^O(param); b \in_R \{0, 1\}; C^* \leftarrow Enc(S^*, c^*, RT^*, m_b); b' \leftarrow A^O(C^*, State)] - \frac{1}{2}|,$$

where State is the state information, $\mathcal{O}_{sk}, \mathcal{O}_{rk}, \mathcal{O}_{re}, \mathcal{O}_{dec}, \mathcal{O}_{decr}$ *are the oracles defined in Definition 2.* $\mathcal{O}_{ts}$ *is the timed-release key extraction oracle that takes as input a release time RT (except for the challenge release time* RT^**) and outputs a timed-release key* τ. *The constraints for* $\mathcal{O}_{dec}$ *and* $\mathcal{O}_{decr}$ *remain the same as those in Definition 2, while there is no restriction for* $\mathcal{O}_{sk}$ *and* $\mathcal{O}_{rk}$. *Besides,* $\mathcal{O}_{re}$ *outputs* $\perp$ *if it is queried on a re-encrypted ciphertext.*

Remark. Note that a static model can be captured if $\mathcal{A}$ is limited to output the challenge before seeing the public parameters.

3 Preliminaries

Bilinear Maps. Let $BSetup$ be an algorithm that on input the security parameter λ, outputs the parameters of a bilinear map as $(q, g, h, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbb{G}_{T'}, e, \bar{e})$,

where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and $\mathbb{G}_{T'}$ are multiplicative cyclic groups of prime order q, where $|q| = \lambda$, and g, h are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The mappings $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and $\bar{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_{T'}$ have three properties: (1) *Bilinearity*: for all $a, b \in_R \mathbb{Z}_q^*$, $e(g^a, h^b) = e(g, h)^{ab}$, $e(g^a, g^b) = e(g, g)^{ab}$; (2) *Non-degeneracy*: $e(g, h) \neq 1_{\mathbb{G}_T}$, $e(g, g) \neq 1_{\mathbb{G}_{T'}}$, where $1_{\mathbb{G}_T}$ and $1_{\mathbb{G}_{T'}}$ are the unit of $\mathbb{G}_T$ and $\mathbb{G}_{T'}$; (3) *Computability*: e and $\bar{e}$ can be efficiently computed.

The General Decisional Diffie-Hellman Exponent Assumption. We review the general decisional Diffie-Hellman exponent problem in the symmetric case so that $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ as in [16]. Let $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow BSetup(1^\lambda)$, and set $g_1 = e(g, g) \in \mathbb{G}_T$, where $\mathbb{G}, \mathbb{G}_T$ are two multiplicative cyclic groups with prime order q and $g \in \mathbb{G}$ is a generator. Let s, n be positive integers and $P, Q \in \mathbb{F}_q[X_1, ..., X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_q$. We write $P = (p_1, ..., p_s)$, $Q = (q_1, ..., q_s)$ and set $p_1 = q_1 = 1$. For any function $h : \mathbb{F}_q \to \Omega$ and vector $(x_1, ..., x_n) \in \mathbb{F}_q^n$, $h(P(x_1, ..., x_n))$ denotes $(h(p_1(x_1, ..., x_n)), ..., h(p_s(x_1, ..., x_n))) \in \Omega^s$. Note that a similar notation can be used for Q. Let $f \in \mathbb{F}_q[X_1, ..., X_n]$. If there exists the following linear decomposition: $f = \sum_{1 \leq i,j \leq s} a_{i,j} p_i p_j + \sum_{1 \leq i \leq s} b_i q_i$, where $a_{i,j}, b_i \in \mathbb{Z}_q$. We say that f depends on (P, Q), i.e. $f \in \langle P, Q \rangle$. The (P, Q, f)-General Decisional Diffie-Hellman Exponent ((P, Q, f)-GDDHE) problem [16] is defined as follows. Note that we let P, Q be as above and $f \in \mathbb{F}_q[X_1, ..., X_n]$.

Definition 4. ***(P,Q,f)-GDDHE Assumption.*** *Given the tuple $H(x_1, ..., x_n) = (g^{P(x_1,...,x_n)}, g_1^{Q(x_1,...,x_n)}) \in \mathbb{G}^s \times \mathbb{G}_T^s$ and $T \in_R \mathbb{G}_T$, the (P,Q,f)-GDDHE problem is to decide whether $T = g_1^{f(x_1,...,x_n)}$. Define $Adv_{\mathcal{A}}^{(P,Q,f)-GDDHE} = |Pr[\mathcal{A}(H(x_1, ..., x_n), g_1^{f(x_1,...,x_n)}) = 0] - Pr[\mathcal{A}(H(x_1, ..., x_n), T) = 0]|$ as the advantage of $\mathcal{A}$ in wining the (P,Q,f)-GDDHE problem. We say that the (P, Q, f)-GDDHE assumption holds in $\mathbb{G}$ if no PPT algorithm has non-negligible advantage.*

Definition 5. ***Bilinear Diffie-Hellman (BDH) Assumption [4].*** *Given the tuple $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, the BDH problem is to compute $e(g, g)^{abc}$, where $a, b, c \in \mathbb{Z}_q^*$. Define $Adv_{\mathcal{A}}^{BDH} = Pr[\mathcal{A}(g, g^a, g^b, g^c) = e(g, g)^{abc}]$ as the advantage of $\mathcal{A}$ in winning the BDH problem. We say that the BDH assumption holds in $\mathbb{G}$ if no PPT algorithm has non-negligible advantage.*

Target Collision Resistant Hash Function. Target Collision Resistant (TCR) hash function was introduced by Cramer and Shoup [15]. A TCR hash function H guarantees that given a random element x which is from the valid domain of H, a PPT adversary $\mathcal{A}$ cannot find $y \neq x$ such that $H(x) = H(y)$. We let $Adv_{H,\mathcal{A}}^{TCR} = Pr[(x, y) \leftarrow \mathcal{A}(1^\lambda) : H(x) = H(y), x \neq y, x, y \in DH]$ be the advantage of $\mathcal{A}$ in successfully finding collisions from a TCR hash function H, where DH is the valid input domain of H. If a hash function is chosen from a TCR hash function family, $Adv_{H,\mathcal{A}}^{TCR}$ is negligible.

4 A New TR-CPBRE Scheme

In this section, we start with a new CPBRE scheme which is considered as a basic scheme for constructing a TR-CPBRE. The new scheme has two building blocks: an IBBE [16] and a TCR hash function [15] where we employ the CHK technique [7] using a TCR hash function to make a "signature" on the ciphertext and including a "verifying key" in the ciphertext. To extend the new scheme to a TR-CPBRE, we employ Boneh and Franklin (BF) IBE scheme [4], and regard a release time as an identity and a timed-release key as the secret key corresponding to the identity. We then use IBE to re-encrypt the ciphertext output such that the decryption of the underlying plaintext requires two pieces of secret information: one is the secret key of the delegatee generated in our CPBRE, the other is the secret key of an identity generated in the IBE scheme.

Our technique is different from that of [14]. We begin with an IBBE scheme which is used to construct a new CCA-secure CPBRE, then we propose a TR-CPBRE by combining the new CPBRE with an IBE. In [14], Chu et al. started with an HICBE [2] and extended the HICBE to a RCCA-secure CPBRE. In the following, we provide the details of our construction.

1. $Setup(1^\lambda, n)$. Let $c \in \{0,1\}^*$ be a condition and $RT \in \{0,1\}^\lambda$ be a release time. Choose $\gamma, \bar{r} \in_R \mathbb{Z}_q^*$, three generators $g, g' \in \mathbb{G}_1$, $h \in \mathbb{G}_2$ and hash functions: $H_0 : \{0,1\}^{2\lambda} \to \mathbb{Z}_q^*$, $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_T \to \{0,1\}^{2\lambda}$, $H_3 : \{0,1\}^* \to \mathbb{G}_1$, $H_4 : \{0,1\}^* \to \mathbb{G}_1$, $H_5 : \{0,1\}^\lambda \to \mathbb{Z}_q^*$, $H_6 : \{0,1\}^* \to \mathbb{G}_1$, $H_7 : \{0,1\}^\lambda \to \mathbb{G}_1$, $H_8 : \mathbb{G}_{T'} \to \{0,1\}^{2\lambda}$, $H_9 : \{0,1\}^{4\lambda} \to \mathbb{Z}_q^*$, $H_{10} : \{0,1\}^{2\lambda} \to \{0,1\}^{2\lambda}$. The master secret key is $msk = (g', \gamma)$, the public key is $param = (g, h, w, v, h^\gamma, \ldots, h^{\gamma^n}, H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8, H_9, H_{10}, TP)$, and the secret key of time server is $sk_{TS} = \bar{r}$, where $w = g'^\gamma$, $v = e(g', h)$ and $TP = g^{\bar{r}}$. Hereafter let s and s' be two maximum numbers of receivers in two identity sets S and $\overline{S}$, respectively, where $s \leq n$, $s' \leq n$.
2. $KeyGen(msk, ID)$. Given $msk = (g', \gamma)$ and an identity ID, output the secret key $sk_{ID} = g'^{\frac{1}{\gamma + H_1(ID)}}$.
3. $TS(sk_{TS}, RT)$. Given sk_{TS} and a release time RT, output a timed-release key $\tau = H_7(RT)^{\bar{r}}$.
4. $Enc(S, c, RT, m)$. Choose $\alpha \in_R \{0,1\}^\lambda, \sigma \in_R \{0,1\}^{2\lambda}$, compute $k = H_0(m,\alpha)$, set $C_1 = w^{-k}$, $C_2 = h^{k \cdot \prod_{i=1}^{s} (\gamma + H_1(ID_i))}$[8], $C_3 = (m||\alpha) \oplus H_2(e(g', h)^k)$, $\bar{C}_3 = C_3 \oplus H_{10}(\sigma)$, $C_4 = H_3(c, S, RT)^k$, $C_5 = H_4(C_1, \bar{C}_3, C_4, C_6, C_7)^k$, $C_6 = g^{\bar{k}}$, $C_7 = \sigma \oplus H_8(\bar{e}(H_7(RT), TP)^{\bar{k}})$, and output $C = (RT, C_1, C_2, \bar{C}_3, C_4, C_5, C_6, C_7)$, where $\bar{k} = H_9(\sigma, C_3)$, $ID_i \in S$, $m \in \{0,1\}^\lambda$.
5. $ReKeyGen(ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$. Choose $\rho \in_R \mathbb{Z}_q^*$, $\{\theta, \alpha'\} \in_R \{0,1\}^\lambda$, compute $k' = H_0(\theta, \alpha')$, $rk_0 = sk_{ID_i}^{H_5(\theta)} \cdot (H_3(c, S, RT)^\rho)$, $rk_1 = w^{-k'}$, $rk_2 = h^{k' \cdot \prod_{i'=1}^{s'} (\gamma + H_1(\overline{ID}_{i'}))}$, $rk_3 = (\theta||\alpha') \oplus H_2(e(g', h)^{k'})$, $rk_4 = H_6(RT, c, rk_1,$

[8] The encryptor can compute C_2 with knowledge of $param$, k and S.

rk_2, $rk_3)^{k'}$, $rk_5 = h^{\rho \cdot \prod_{i=1}^{s} (\gamma + H_1(ID_i))}$, and output the re-encryption key $rk_{ID_i \to \overline{S}|RT,c} = (rk_0, rk_1, rk_2, rk_3, rk_4, rk_5)$, where $ID_i \in S$, $\overline{ID}_{i'} \in \overline{S}$.

6. $ReEnc(rk_{ID_i \to \overline{S}|RT,c}, ID_i, S, \overline{S}, c, RT, C)$.
   (1) Verify the validity of original ciphertext C

$$\begin{aligned} & e(w^{-1}, C_2) \stackrel{?}{=} e(C_1, h^{\prod_{i=1}^{s} (\gamma + H_1(ID_i))}), \\ & \bar{e}(w^{-1}, C_4) \stackrel{?}{=} \bar{e}(C_1, H_3(c, S, RT)), \\ & \bar{e}(w^{-1}, C_5) \stackrel{?}{=} \bar{e}(C_1, H_4(C_1, \bar{C}_3, C_4, C_6, C_7)), ID_i \stackrel{?}{\in} S. \end{aligned} \tag{1}$$

   If Eq. (1) does not hold, output $\perp$. Otherwise, proceed.
   (2) Compute $C_2' = e(rk_0, C_2)/e(C_4, rk_5)$, output $C_R = (RT, C_1, C_2', \bar{C}_3, C_4, C_6, C_7, rk_1, rk_2, rk_3, rk_4)$.
7. $Dec(sk_{ID_i}, ID_i, S, c, RT, C, \tau)$.
   (1) Verify Eq. (1). If the equation does not hold, output $\perp$. Otherwise, proceed.
   (2) Compute $\sigma = C_7 \oplus H_8(\bar{e}(\tau, C_6))$, $C_3 = \bar{C}_3 \oplus H_{10}(\sigma)$, $e(g', h)^k = (e(C_1, h^{B_{i,s(\gamma)}}) e(sk_{ID_i}, C_2))^{\beta}$, and $m \| \alpha = C_3 \oplus H_2(e(g', h)^k)$, where $\beta = \frac{1}{\prod_{j=1, j \neq i}^{s} H_1(ID_j)}$, and $B_{i,s(\gamma)} = \frac{1}{\gamma} \cdot (\prod_{j=1, j \neq i}^{s} (\gamma + H_1(ID_j)) - \prod_{j=1, j \neq i}^{s} H_1(ID_j))$[9]. If $C_6 = g^{H_9(\sigma, C_3)}$ and $C_1 = w^{-H_0(m, \alpha)}$, output m. Otherwise, output $\perp$.
8. $Dec_R(sk_{\overline{ID}_{i'}}, ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R, \tau)$.
   (1) Compute $e(g', h)^{k'} = (e(rk_1, h^{B_{i', s'(\gamma)}}) e(sk_{\overline{ID}_{i'}}, rk_2))^{\beta'}$, and $\theta \| \alpha' = rk_3 \oplus H_2(e(g', h)^{k'})$, where $\beta' = \frac{1}{\prod_{j'=1, j' \neq i'}^{s'} H_1(\overline{ID}_{j'})}$, and $B_{i', s'(\gamma)} = \frac{1}{\gamma} \cdot (\prod_{j'=1, j' \neq i'}^{s'} (\gamma + H_1(\overline{ID}_{j'})) - \prod_{j'=1, j' \neq i'}^{s'} H_1(\overline{ID}_{j'}))$.
   (2) Verify

$$\begin{aligned} & rk_1 \stackrel{?}{=} w^{-H_0(\theta, \alpha')}, rk_2 \stackrel{?}{=} h^{H_0(\theta, \alpha') \cdot \prod_{i'=1}^{s'} (\gamma + H_1(\overline{ID}_{i'}))}, \\ & rk_4 \stackrel{?}{=} H_6(RT, c, rk_1, rk_2, rk_3)^{H_0(\theta, \alpha')}, \overline{ID}_{i'} \stackrel{?}{\in} \overline{S}. \end{aligned} \tag{2}$$

   If Eq. (2) does not hold, output $\perp$. Otherwise, proceed.
   (3) Compute $\sigma = C_7 \oplus H_8(\bar{e}(\tau, C_6))$ and $C_3 = \bar{C}_3 \oplus H_{10}(\sigma)$. If $ID_i \in S$, compute $M = (e(C_1, h^{B_{i,s(\gamma)}}) (C_2')^{H_5(\theta)^{-1}})^{\beta}, m || \alpha = C_3 \oplus H_2(M)$, where $\beta = \frac{1}{\prod_{j=1, j \neq i}^{s} H_1(ID_j)}, B_{i,s(\gamma)} = \frac{1}{\gamma} \cdot (\prod_{j=1, j \neq i}^{s} (\gamma + H_1(ID_j)) -$

[9] With knowledge of $param$, ID_i and S, the decryptor is able to compute $h^{B_{i,s(\gamma)}}$. More details can be found in [16].

$\prod_{j=1,j\neq i}^{s} H_1(ID_j)$). If $C_1 = w^{-H_0(m,\alpha)}, C_4 = H_3(c, S, RT)^{H_0(m,\alpha)}$, and $C_6 = g^{H_9(\sigma,C_3)}$, output m. Otherwise, output $\perp$.

Correctness: It is easy to verify that the underlying plaintexts of the original and re-encrypted ciphertexts can be recovered correctly if the ciphertexts are computed via the description above. We hence skip the details.

Theorem 1. *Suppose (P, Q, f)-GDDHE assumption holds, our TR-CPBRE scheme for n receivers is IND-sID-CCA-secure at original ciphertext in the random oracle model. If there is a PPT adversary $\mathcal{A}$, who issues at most q_{H_i} queries to H_i and breaks the $(\bar{t}, q_{sk}, q_{rk}, q_{re}, q_{d2}, q_{d1}, \epsilon)$-IND-sID-CCA-Or security of our TR-CPBRE scheme, then we can construct a PPT adversary $\mathcal{C}$ to solve the (t', ϵ')-(P, Q, f)-GDDHE problem with*

$$\epsilon' \geq \frac{1}{q_{H_2}}(\epsilon - \frac{q_{H_0} + (q_{H_0} + q_{H_2})(q_{d_1} + q_{d_2})}{2^{2\lambda}} - \frac{2(q_{d_1} + q_{d_2}) + q_{re}}{q}),$$

$$\begin{aligned} t' \leq \bar{t} + O(1)(q_{H_i} + q_{sk} + q_{rk} + q_{re} + q_{d2} + q_{d1}) + t_e(q_{sk} + q_{H_0}(2q_{re} + 2q_{d2} \\ + 8q_{d1}) + (2n + 9)q_{rk} + (3n + 8)q_{re} + (n + 2)q_{d2} + (2n + 4)q_{d1}) \\ + t_p(7q_{re} + 7q_{d2} + 2q_{d1}), \end{aligned}$$

where t_e denotes the running time of an exponentiation, t_p denotes the running time of a pairing, q_{sk}, q_{rk}, q_{re}, q_{d2}, q_{d1} denote the total number of secret key extraction queries, re-encryption key extraction queries, re-encryption queries, the original and re-encrypted ciphertexts decryption queries, respectively, $i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

Please refer to Appendix A for the proof of Theorem 1.

Theorem 2. *Suppose the BDH assumption holds and H_7, H_8, H_9, H_{10} are TCR hash functions, our TR-CPBRE scheme for n receivers is IND-CTCA-secure at original ciphertext.*

Please refer to Appendix B for the proof of Theorem 2.

5 Comparison

In this section we compare our scheme with [14] and [17] in terms of property and efficiency. Two tables are listed in the following: Table 2 shows the comparison of computational cost, and Table 3 shows the comparison of communication complexity. We first define the notations and parameters used in the Tables. Let $|\mathbb{G}_R|$ and $|\mathbb{G}_t|$ denote the bit-length of an element in group $\mathbb{G}_1$ or $\mathbb{G}_2$, $\mathbb{G}_T$ or $\mathbb{G}_{T'}$, $|svk|$ and $|\sigma|$ denote the bit-length of the verification key and signature of one time signature, λ denote the security parameter, t_p, t_e, t_v, t_s denote the computation cost of a bilinear pairing, an exponentiation, one verification and one signature of a one-time signature, respectively. Suppose [14], [17] and our scheme share the same number (N) of delegatees for each delegation; meanwhile,

assume the release time and identity set of the above schemes are the public information such that they could be precluded from the ciphertext.

Table 1 generally shows that our scheme achieves all properties with CCA security under the (P, Q, f)-GDDHE assumption. Specifically, [14] is RCCA secure under the decisional n-$BDHE$ assumption [5], whereas our scheme is CCA secure and additionally supports timed-release property. Compared to [17], which is secure against RCCA under the 3-$QDBDH$ assumption [21], our scheme provides conditional delegation and broadcast re-encryption without losing CCA security. In conclusion, our scheme enables the delegator to implement a more fine-grained delegation of decryption rights in cloud storage systems where CCA security is required, but its security relies on random oracles. The problem of proposing a TR-CPBRE scheme with CCA security in the standard model remains open.

Table 2. Computation Cost Comparison

Schemes	**Computation Cost**				
	Enc	$ReEnc$	Dec	Dec_R	$ReKey$
CPBRE [14]	$(N+6)t_e + t_p$	$(N+3)t_e + 6t_p$	$2Nt_e + 8t_p$	$(4N-1)t_e + 20t_p$	$(N+7)t_e + t_p$
TR-PRE [17]	$t_s + 10t_e + 4t_p$	$(4t_e + t_v)N + 2Nt_p$	$5t_e + t_v + 5t_p$	$5t_e + t_v + 7t_p$	Nt_e
Our scheme	$(N+8)t_e + 2t_p$	$(N+1)t_e + 8t_p$	$(2N+4)t_e + 9t_p$	$(3N+9)t_e + 4t_p$	$(2N+9)t_e + t_p$

From Table 2, we see that [14] suffers from the largest number of pairings and [17] suffers from $O(N)$ complexity in pairings. Compared with [14], our TR-CPBRE requires one additional t_p in Enc and Dec, and $2t_p$ in $ReEnc$, respectively, but significantly reduces the number of pairings ($16t_p$) in Dec_R without requiring more pairings with regard to $ReKey$. As opposed to [17], our scheme achieves a constant cost of parings in $ReEnc$, and has the same number of pairings in the sum of cost of other metrics (except for $ReKey$). In conclusion, our scheme requires less number of pairings when compared with [14] and [17].

Table 3. Communication Cost Comparison

Schemes	**Ciphertexts and Re-Encryption Key Length**		
	Re-Encrypted Ciphertext	Original Ciphertext	ReKey
CPBRE [14]	$10\|\mathbb{G}_R\| + 2\|\mathbb{G}_t\|$	$4\|\mathbb{G}_R\| + \|\mathbb{G}_t\|$	$5\|\mathbb{G}_R\| + \|\mathbb{G}_t\|$
TR-PRE [17]	$(N+4)\|\mathbb{G}_R\| + 3\|\mathbb{G}_t\| + \|svk\| + \|\sigma\|$	$3\|\mathbb{G}_R\| + 3\|\mathbb{G}_t\| + \|svk\| + \|\sigma\|)$	$N\|\mathbb{G}_R\|$
Our scheme	$5\|\mathbb{G}_R\| + 2\|2\lambda\|$	$6\|\mathbb{G}_R\| + \|\mathbb{G}_t\| + 3\|2\lambda\|$	$5\|\mathbb{G}_R\| + \|2\lambda\|$

Table 3 shows that [14] achieves the smallest number of group elements in ciphertext, while [17] suffers from linear cost in ciphertext and $ReKey$. When compared with [14], our scheme reduces 3 elements in $\mathbb{G}_R$ and $\mathbb{G}_t$, respectively. The number of group elements of our scheme in ciphertext is less than that of [17]; meanwhile, the cost of our scheme is constant. In $ReKey$ our TR-CPBRE and [14] need one additional group of elements compared to [17]. It is worth mentioning that, when compared with [17], [14] and our scheme enjoy better efficiency in communication as N increases.

6 Concluding Remarks

In this paper, we introduced a new variant of PRE, named TR-CPBRE, which achieves conditional delegation, broadcast re-encryption and timed-release property simultaneously. To the best of our knowledge, our TR-CPBRE is the first of its kind. We also showed that our scheme can be proved IND-sID-CCA secure in the random oracle model under the (P, Q, f)-GDDHE assumption. Moreover, when compared with the existing CPBRE and TR-PRE schemes, our scheme not only requires less number of pairings and achieves better efficiency in communication, but also enables the delegator to make a fine-grained delegation of decryption rights to multiple delegatees without losing CCA security.

This paper also motivates some interesting open problems, for example, how to construct a CCA-secure TR-CPBRE scheme in the adaptive identity model, i.e. achieving IND-aID-CCA security.

References

1. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM TISSEC 9(1), 1–30 (2006)
2. Attrapadung, N., Furukawa, J., Imai, H.: Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In: ASIACRYPT '06. LNCS, vol. 4284, pp. 161–177. Springer (2006)
3. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: EUROCRYPT '98. pp. 127–144. Springer (1998)
4. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: CRYPTO '01. LNCS, vol. 2139, pp. 213–229. Springer (2001)
5. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) Advances in Cryptology CRYPTO 2005, LNCS, vol. 3621, pp. 258–275. Springer (2005), `http://dx.doi.org/10.1007/11535218_16`
6. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 211–220. CCS '06, ACM, New York, NY, USA (2006), `http://doi.acm.org/10.1145/1180405.1180432`
7. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Eurocrypt '04. LNCS, vol. 3027, pp. 207–222. Springer (2004)

8. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 185–194. CCS '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1315245.1315269
9. Canetti, R., Krawczyk, H., Nielsen, J.: Relaxing chosen-ciphertext security. In: CRYPTO '03. LNCS, vol. 2729, pp. 565–582. Springer (2003)
10. Cathalo, J., Libert, B., Quisquater, J.J.: Efficient and non-interactive timed-release encryption. In: Qing, S., Mao, W., Lpez, J., Wang, G. (eds.) Information and Communications Security, LNCS, vol. 3783, pp. 291–303. Springer (2005), http://dx.doi.org/10.1007/11602897_25
11. Chalkias, K., Hristu-Varsakelis, D., Stephanides, G.: Improved anonymous timed-release encryption. In: Biskup, J., Lpez, J. (eds.) Computer Security ESORICS 2007, LNCS, vol. 4734, pp. 311–326. Springer (2007), http://dx.doi.org/10.1007/978-3-540-74835-9_21
12. Cheon, J.H., Hopper, N., Kim, Y., Osipkov, I.: Provably secure timed-release public key encryption. ACM TISS. 11(2) (2008)
13. Chow, S.S., Yiu, S.: Timed-release encryption revisited. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) Provable Security, LNCS, vol. 5324, pp. 38–51. Springer (2008), http://dx.doi.org/10.1007/978-3-540-88733-1_3
14. Chu, C.K., Weng, J., Chow, S.S., Zhou, J., Deng, R.H.: Conditional proxy broadcast re-encryption. In: ACISP '09. LNCS, vol. 5594, pp. 327–342. Springer (2009)
15. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. 33(1), 167–226 (January 2004)
16. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: ASIACRYPT '07. LNCS, vol. 4833, pp. 200–215. Springer (2007)
17. Emura, K., Miyaji, A., Omote, K.: A timed-release proxy re-encryption scheme. IEICE Transactions 98-A(8), 1682–1695 (2011)
18. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D. (ed.) Advances in Cryptology CRYPTO 93, LNCS, vol. 773, pp. 480–491. Springer (1994), http://dx.doi.org/10.1007/3-540-48329-2_40
19. Hanaoka, G., Kawai, Y., Kunihiro, N., Matsuda, T., Weng, J., Zhang, R., Zhao, Y.: Generic construction of chosen ciphertext secure proxy re-encryption. In: CT-RSA '12. LNCS, vol. 7178, pp. 349–364. Springer (2012)
20. Kikuchi, R., Fujioka, A., Okamoto, Y., Saito, T.: Strong security notions for timed-release public-key encryption revisited. In: Kim, H. (ed.) Information Security and Cryptology - ICISC 2011, LNCS, vol. 7259, pp. 88–108. Springer (2012), http://dx.doi.org/10.1007/978-3-642-31912-9_7
21. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: PKC'08. PKC'08, vol. 4939, pp. 360–379. Springer (2008)
22. Matsuda, T., Nakai, Y., Matsuura, K.: Efficient generic constructions of timed-release encryption with pre-open capability. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing-Based Cryptography - Pairing 2010, LNCS, vol. 6487, pp. 225–245. Springer (2010), http://dx.doi.org/10.1007/978-3-642-17455-1_15
23. May, T.: Timed-release cryptography (February, 1993), unpublished manuscript, available at http://www.hks.net.cpunks/cpunks-0/1560.html
24. Nakai, Y., Matsuda, T., Kitada, W., Matsuura, K.: A generic construction of timed-release encryption with pre-open capability. In: Takagi, T., Mambo, M. (eds.) Advances in Information and Computer Security, LNCS, vol. 5824, pp. 53–70. Springer (2009), http://dx.doi.org/10.1007/978-3-642-04846-3_5

25. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Tech. rep., Cambridge, MA, USA (1996)

A Proof of Theorem 1

Proof. We prove our scheme in the static notion of Definition 2, that is, an adversary $\mathcal{A}$ is restricted to output a challenge identity, a challenge release time and a challenge condition before the setup phase. Suppose there exists the adversary $\mathcal{A}$ who can break the IND-sID-CCA-Or security of TR-CPBRE. We then construct a reduction algorithm $\mathcal{C}$ to decide whether $T = g_1^{f(x_1,...,x_n)}$ or $T \in_R \mathbb{G}_T$. $\mathcal{C}$ is given as input a (f, g, F)-GDDHE instance, i.e., $(g_0, g_0^{\gamma}, ..., g_0^{\gamma^{t-1}}, g_0^{\gamma \cdot f(\gamma)}, g_0^{k \cdot \gamma \cdot f(\gamma)}, h_0, h_0^{\gamma}, ..., h_0^{\gamma^{2n}}, h_0^{k \cdot g(\gamma)})$ and $T \in \mathbb{G}_T$, where g_0 is a generator of $\mathbb{G}_1$, h_0 is a generator of $\mathbb{G}_2$, f and g are two coprime polynomials with pairwise distinct roots, of respective orders t (which is the number of the secret keys generated in the simulation) and n (which is the maximum size of a set of receivers), T is either equal to $e(g_0, h_0)^{k \cdot f(\gamma)}$ or to $T' \in_R \mathbb{G}_T$. Here we define some notations for polynomials used in the following proof. We denote $f(X) = \prod_{i=1}^{t} (X + x_i)$, $g(X) = \prod_{i=t+1}^{t+n} (X + x_i)$, $f_i(x) = \frac{f(x)}{x+x_i}$, $g_j(x) = \frac{g(x)}{x+x_j}$, where $i \in \{1, ..., t\}$, $j \in \{t+1, ..., t+n\}$, deg $f_i(x) = t-1$ and deg $g_j(x) = n-1$.

1. **Initialization.** $\mathcal{A}$ outputs $S^* = \{ID_1^*, ..., ID_s^*\}$, c^* and RT^*, where $s \leq n$.
2. **Setup.** $\mathcal{C}$ sets $g' = g_0^{f(\gamma)}$ (g' cannot be computed by $\mathcal{C}$), $h = h_0^{\prod_{i=t+s+1}^{t+n} (\gamma + x_i)}$, $w = g_0^{\gamma \cdot f(\gamma)} = g'^{\gamma}$ and $v = e(g_0, h_0)^{f(\gamma) \cdot \prod_{i=t+s+1}^{t+n} (\gamma + x_i)} = e(g', h)$, chooses a generator $g \in_R \mathbb{G}_1$ and a $\bar{r} \in_R \mathbb{Z}_q^*$, computes $TP = g^{\bar{r}}$, and sends $param = (g, h, w, v, h^{\gamma}, ..., h^{\gamma^n}, H_j, TP)$ and $\bar{r}$ to $\mathcal{A}$, where $j \in \{0, ..., 10\}$ and $x_i = H_1(ID_i), i \in \{t+s+1, ..., t+n\}$. At any time, $\mathcal{A}$ can adaptively query the random oracles H_j ($j \in \{0, ..., 10\}$) which are controlled by $\mathcal{C}$. $\mathcal{C}$ maintains the lists H_j^{List} ($j \in \{0, ..., 10\}$) which are initially empty (except for H_1^{List}) and answers the queries as follows.
   (a) H_0: on receipt of an H_0 query on (m, α), if there is a tuple (m, α, k) in H_0^{List}, $\mathcal{C}$ forwards the predefined value k to $\mathcal{A}$, where $k \in \mathbb{Z}_q^*$. Otherwise, $\mathcal{C}$ sets $H_0(m, \alpha) = k$, responds k to $\mathcal{A}$ and adds the tuple (m, α, k) to H_0^{List}, where $k \in_R \mathbb{Z}_q^*$.
   (b) H_1: on receipt of an H_1 query on $ID_i \in \{0,1\}^*$, if there is a tuple (ID_i, x_i) in H_1^{List}, $\mathcal{C}$ forwards the predefined value x_i to $\mathcal{A}$, where $x_i \in \mathbb{Z}_q^*$. Otherwise, $\mathcal{C}$ sets $H_1(ID_i) = x_i$, responds x_i to $\mathcal{A}$ and adds the tuple (ID_i, x_i) to H_1^{List}, where $x_i \in_R \mathbb{Z}_q^*$. Note that H_1 contains $\{(*, x_i)\}_{i=1}^{t}, \{(ID_i^*, x_i)\}_{i=t+1}^{t+s}$ at the beginning, where $ID_i^* \in S^*$, $*$ is an empty entry in H_1^{List}.
   (c) H_2: on receipt of an H_2 query on $R \in \mathbb{G}_T$, if there is a tuple (R, δ_1) in H_2^{List}, $\mathcal{C}$ forwards the predefined value δ_1 to $\mathcal{A}$, where $\delta_1 \in \{0,1\}^{2\lambda}$. Otherwise, $\mathcal{C}$ sets $H_2(R) = \delta_1$, responds δ_1 to $\mathcal{A}$ and adds the tuple (R, δ_1) to H_2^{List}, where $\delta_1 \in_R \{0,1\}^{2\lambda}$.

(d) H_3: on receipt of an H_3 query on (c, S, RT), if there is a tuple (c, RT, S, δ_2, ξ_1) in H_3^{List}, $\mathcal{C}$ forwards the predefined value ξ_1 to $\mathcal{A}$, where $\xi_1 \in \mathbb{G}_1$. Otherwise, $\mathcal{C}$ sets $\xi_1 = g_0^{\gamma \cdot f(\gamma) \cdot \delta_2}$. $\mathcal{C}$ then responds ξ_1 to $\mathcal{A}$ and adds the tuple $(c, RT, S, \delta_2, \xi_1)$ to H_3^{List}.

(e) H_4: on receipt of an H_4 query on (C_1, $\bar{C}_3$, C_4, C_6, C_7), if there is a tuple $(C_1, \bar{C}_3, C_4, C_6, C_7, \delta_3, \xi_2)$ in H_4^{List}, $\mathcal{C}$ forwards the predefined value ξ_2 to $\mathcal{A}$, where $\xi_2 \in \mathbb{G}_1$. Otherwise, $\mathcal{C}$ sets $\xi_2 = g_0^{\gamma \cdot f(\gamma) \cdot \delta_3}$, responds ξ_2 to $\mathcal{A}$ and adds the tuple $(C_1, \bar{C}_3, C_4, C_6, C_7, \delta_3, \xi_2)$ to H_4^{List}, where $\delta_3 \in_R \mathbb{Z}_q^*$.

(f) H_5: on receipt of an H_5 query on $\theta \in \{0,1\}^\lambda$, if there is a tuple (θ, δ_4) in H_5^{List}, $\mathcal{C}$ forwards the predefined value δ_4 to $\mathcal{A}$, where $\delta_4 \in \mathbb{Z}_q^*$. Otherwise, $\mathcal{C}$ sets $H_5(\theta) = \delta_4$, responds δ_4 to $\mathcal{A}$ and adds the tuple (θ, δ_4) to H_5^{List}, where $\delta_4 \in_R \mathbb{Z}_q^*$.

(g) H_6: on receipt of an H_6 query on $(RT, c, rk_1, rk_2, rk_3)$, if there is a tuple $(RT, c, rk_1, rk_2, rk_3, \delta_5, \xi_3)$ in H_6^{List}, $\mathcal{C}$ forwards the predefined value ξ_3 to $\mathcal{A}$, where $\xi_3 \in \mathbb{G}_1$. Otherwise, $\mathcal{C}$ sets $\xi_3 = g_0^{\delta_5}$, where $\delta_5 \in_R \mathbb{Z}_q^*$. Then, $\mathcal{C}$ responds ξ_3 to $\mathcal{A}$ and adds the tuple $(RT, c, rk_1, rk_2, rk_3, \delta_5, \xi_3)$ to H_6^{List}.

(h) H_7: on receipt of an H_7 query on $RT \in \{0,1\}^\lambda$, if there is a tuple (RT, δ_6, ξ_4) in H_7^{List}, $\mathcal{C}$ forwards the predefined value ξ_4 to $\mathcal{A}$, where $\xi_4 \in \mathbb{G}_1$. Otherwise, $\mathcal{C}$ sets $\xi_4 = g_0^{\delta_6}$, where $\delta_6 \in_R \mathbb{Z}_q^*$, responds ξ_4 to $\mathcal{A}$ and adds the tuple (RT, δ_6, ξ_4) to H_7^{List}.

(i) H_8: on receipt of an H_8 query on $\bar{R} \in \mathbb{G}_{T'}$, if there is a tuple $(\bar{R}, \delta_7)$ in H_8^{List}, $\mathcal{C}$ forwards the predefined value δ_7 to $\mathcal{A}$, where $\delta_7 \in \{0,1\}^{2\lambda}$. Otherwise, $\mathcal{C}$ sets $H_8(\bar{R}) = \delta_7$, responds δ_7 to $\mathcal{A}$ and adds the tuple $(\bar{R}, \delta_7)$ to H_8^{List}, where $\delta_7 \in_R \{0,1\}^{2\lambda}$.

(j) H_9: on receipt of an H_9 query on (σ, C_3), if there is a tuple $(\sigma, C_3, \bar{k})$ in H_9^{List}, $\mathcal{C}$ forwards the predefined value $\bar{k}$ to $\mathcal{A}$, where $\bar{k} \in \mathbb{Z}_q^*$. Otherwise, $\mathcal{C}$ sets $H_9(\sigma, C_3) = \bar{k}$, responds $\bar{k}$ to $\mathcal{A}$ and adds the tuple $(\sigma, C_3, \bar{k})$ to H_9^{List}, where $\bar{k} \in_R \mathbb{Z}_q^*$.

(k) H_{10}: on receipt of an H_{10} query on $\sigma \in \{0,1\}^{2\lambda}$, if there is a tuple (σ, δ_8) in H_{10}^{List}, $\mathcal{C}$ forwards the predefined value δ_8 to $\mathcal{A}$, where $\delta_8 \in \{0,1\}^{2\lambda}$. Otherwise, $\mathcal{C}$ sets $H_{10}(\sigma) = \delta_8$, responds δ_8 to $\mathcal{A}$ and adds the tuple (σ, δ_8) to H_{10}^{List}, where $\delta_8 \in_R \{0,1\}^{2\lambda}$.

In addition, $\mathcal{C}$ also maintains the following lists which are initially empty.

(a) SKT: records the tuples (ID_i, sk_{ID_i}), which are the results of the queries to $\mathcal{O}_{sk}(ID_i)$.

(b) RKT: records the tuples (RT, c, ID_i, S, $\overline{S}$, rk_0, rk_5, C_θ, θ, tag_1, tag_2, tag_3), which are the results of the queries to $\mathcal{O}_{rk}(ID_i, S, \overline{S}, c, RT)$, where tag_1, tag_2, tag_3 denote that the re-encryption key is randomly chosen, generated in $\mathcal{O}_{re}$ or in $\mathcal{O}_{rk}$, respectively. Note that $rk_{ID_i \to \overline{S}|RT,c} = (rk_0, C_\theta, rk_5)$, where $C_\theta = (rk_1, rk_2, rk_3, rk_4)$.

(c) RET: records the tuples $(C_R, ID_i, S, \overline{S}, c, RT, tag_1, tag_2, tag_3)$, which are the results of the queries to $\mathcal{O}_{re}(ID_i, S, \overline{S}, c, RT, C)$, where tag_1, tag_2, tag_3 denote that C_R is re-encrypted under a valid re-encryption key, under a random re-encryption key or generated without any re-encryption key.

3. **Phase I.** $\mathcal{A}$ issues a series of queries to which $\mathcal{C}$ responds as follows.
   (a) $\mathcal{O}_{sk}(ID_i)$: $\mathcal{C}$ generates the secret key sk_{ID_i} for $\mathcal{A}$ as follows:
   - If $ID_i \in S^*$, $\mathcal{C}$ outputs a random bit in $\{0,1\}$ and aborts the simulation due to the restriction defined in Definition 2.
   - Otherwise,
     i. If $(ID_i, sk_{ID_i}) \in SKT$, $\mathcal{C}$ returns sk_{ID_i} to $\mathcal{A}$.
     ii. Otherwise, $\mathcal{C}$ checks whether there is a tuple (ID_i, x_i) in H_1^{List}. If yes, $\mathcal{C}$ generates the secret key $sk_{ID_i} = g_0^{f_i(\gamma)}$, and sends sk_{ID_i} to $\mathcal{A}$. The secret key is valid since $g_0^{f_i(\gamma)} = g_0^{\frac{f(\gamma)}{\gamma+x_i}} = g'^{\frac{1}{\gamma+H_1(ID_i)}}$. Then, $\mathcal{C}$ adds (ID_i, sk_{ID_i}) to SKT. Otherwise, $\mathcal{C}$ first sets $H_1(ID_i) = x_i$, next generates and sends the secret key sk_{ID_i} to $\mathcal{A}$ exactly as above, where $x_i \in_R \mathbb{Z}_q^*$. Finally, $\mathcal{C}$ completes SKT and H_1^{List} for (ID_i, sk_{ID_i}) and (ID_i, x_i), respectively.

   (b) $\mathcal{O}_{rk}(ID_i, S, \overline{S}, c, RT)$: If there is a tuple $(RT, c, ID_i, S, \overline{S}, rk_0, rk_5, C_\theta, \theta, *, 0, 1)$ in RKT, $\mathcal{C}$ returns $rk_{ID_i \to \overline{S}|RT,c}$ to $\mathcal{A}$. Otherwise,
   - If $(ID_i, sk_{ID_i}) \in SKT$, $\mathcal{C}$ first checks whether there is a tuple $(RT, c, ID_i, S, \overline{S}, rk_0, rk_5, C_\theta, \theta, 0, 1, 0)$ in RKT, where $ID_i \in S$. If yes, $\mathcal{C}$ responds $rk_{ID_i \to \overline{S}|RT,c}$ to $\mathcal{A}$ and sets $tag_1 = 0$, $tag_2 = 0$, $tag_3 = 1$. Otherwise, $\mathcal{C}$ first chooses $\theta \in_R \{0,1\}^\lambda$, $\alpha' \in_R \{0,1\}^\lambda$, $\rho \in_R \mathbb{Z}_q^*$, computes $rk_0 = g_0^{f_i(\gamma)\cdot\delta_4} \cdot \xi_1^\rho$ and $rk_5 = h^{\rho \cdot \prod_{i=1}^{\bar{s}} (\gamma+x_i)}$, where $\delta_4 = H_5(\theta)$, $\xi_1 = g_0^{\delta_2}$, $x_i = H_1(ID_i)$ and $\bar{s} = |S|$. Here rk_0 is valid since $rk_0 = g_0^{f_i(\gamma)\cdot\delta_4} \cdot g_0^{\delta_2\cdot\rho} = g'^{\frac{H_5(\theta)}{(\gamma+H_1(ID_i))}} \cdot (H_3(c,S,RT)^\rho)$. Then, $\mathcal{C}$ computes C_θ as in the real scheme, sends the re-encryption key to $\mathcal{A}$ and adds $(RT, c, ID_i, S, \overline{S}, rk_0, rk_5, C_\theta, \theta, 0, 0, 1)$ to RKT.
   - Otherwise:
     i. If $ID_i \in S^* \wedge (\overline{ID}_{i'}, sk_{\overline{ID}_{i'}}) \in SKT$, $\mathcal{C}$ outputs a random bit and aborts the simulation, where $\overline{ID}_{i'} \in \overline{S}$.[10]
     ii. If $ID_i \in S^* \wedge (\overline{ID}_{i'}, sk_{\overline{ID}_{i'}}) \notin SKT$, $\mathcal{C}$ checks whether there is a tuple $(RT, c, ID_i, S, \overline{S}, rk_0, rk_5, C_\theta, \theta, 1, 1, 0)$ in RKT. If yes, $\mathcal{C}$ returns the key $rk_{ID_i \to \overline{S}|RT,c}$ to $\mathcal{A}$ and sets $tag_1 = 1$, $tag_2 = 0$, $tag_3 = 1$. Otherwise, $\mathcal{C}$ chooses $\theta, \alpha' \in_R \{0,1\}^\lambda$ and computes C_θ as in the real scheme. Then, $\mathcal{C}$ chooses a generator $\bar{g_1} \in_R \mathbb{G}_1$ and a $\rho \in_R \mathbb{Z}_q^*$, computes $\delta_4 = H_5(\theta)$ and $rk_0 = \bar{g_1}^{\delta_4} \xi_1^\rho$, $rk_5 = h^{\rho \cdot \prod_{i=1}^{\bar{s}} (\gamma+x_i)}$, where $x_i = H_1(ID_i)$, $\xi_1 = g_0^{\gamma \cdot f(\gamma)\cdot\delta_2}$ and $\bar{s} = |S|$. $\mathcal{C}$ sends the re-encryption key to $\mathcal{A}$ and adds $(RT, c, ID_i, S, \overline{S}, rk_0, rk_5, C_\theta, \theta, 1, 0, 1)$ to RKT.

[10] Note that our proof can be regarded as a weaker notion of Definition 2 as $\mathcal{A}$ is not allowed to achieve any re-encryption key from a challenge identity to a corrupted identity set under a non-challenge condition. However, it does not reduce the advantage of $\mathcal{A}$ in guessing the value of the bit b because $\mathcal{A}$ is able to achieve the corresponding re-encryption output by issuing $\mathcal{O}_{re}$.

iii. Otherwise, $\mathcal{C}$ generates the re-encryption key as in the real scheme.

(c) $\mathcal{O}_{re}(ID_i, S, \overline{S}, c, RT, C)$: $\mathcal{C}$ checks whether Eq. (1) holds. If not, output $\perp$. Else,
- If $(ID_i \in S^* \wedge (\overline{ID}_{i'}, sk_{\overline{ID}_{i'}}) \in SKT)$ does not hold, $\mathcal{C}$ constructs the re-encryption key as step (b) and proceeds.
  i. If $(ID_i, sk_{ID_i}) \in SKT$, $\mathcal{C}$ adds $(RT, c, ID_i, S, \overline{S}, rk_0, rk_5, C_\theta, \theta, 0, 1, 0)$ and $(C_R, ID_i, S, \overline{S}, c, RT, 1, 0, 0)$ to RKT and RET, respectively.
  ii. Otherwise, $\mathcal{C}$ adds $(RT, c, ID_i, S, \overline{S}, rk_0, rk_5, C_\theta, \theta, 1, 1, 0)$ and $(C_R, ID_i, S, \overline{S}, c, RT, 0, 1, 0)$ to RKT and RET, respectively.
- Otherwise, $\mathcal{C}$ checks whether there is a tuple (m, α, k) in H_0^{List} such that $C_1 = w^{-k}$. If no such tuple exists, $\mathcal{C}$ outputs $\perp$. Otherwise, $\mathcal{C}$ checks whether there is a tuple $(RT^*, c^*, ID_i, S^*, \overline{S}, \perp, \perp, C_\theta, \theta, \perp, 1, \perp)$ in RKT. If no, $\mathcal{C}$ choses a new $\theta \in_R \{0,1\}^\lambda$, generates C_θ under $\overline{S}$ as in the real scheme and computes C_2' as $C_2' = (\frac{v^{k \cdot \beta^{-1}}}{e(g_0^{-\gamma \cdot f(\gamma)}, h^{B_{i,\bar{s}(\gamma)}})^k})^{H_5(\theta)}$, where $\bar{s} = |S|$, $\beta = \frac{1}{\prod_{j=1, j \neq i}^{\bar{s}} H_1(ID_j)}$. Finally, $\mathcal{C}$ outputs C_R to $\mathcal{A}$ and adds $(RT^*, c^*, ID_i, S^*, \overline{S}, \perp, \perp, C_\theta, \theta, \perp, 1, \perp)$ and $(C_R, ID_i, S, \overline{S}, c, RT, 0, 0, 1)$ to RKT and RET, respectively.

(d) $\mathcal{O}_{dec}(ID_i, S, c, RT, C)$: $\mathcal{C}$ first verifies whether Eq. (1) holds. If not, $\mathcal{C}$ outputs $\perp$. Otherwise, $\mathcal{C}$ recovers the plaintext m as follows.
- If $(ID_i, sk_{ID_i}) \in SKT$, $\mathcal{C}$ recovers m using sk_{ID_i} as in the real scheme.
- Otherwise, $\mathcal{C}$ checks whether the tuple (m, α, k) is in H_0^{List} and the tuple (R, δ_1) is in H_2^{List} such that $C_1 = w^{-k}$, $\bar{C_3} \oplus H_{10}(\sigma) = (m||\alpha) \oplus \delta_1$, and $R = e(g', h)^k$. Recall that $\mathcal{C}$ can compute τ, and recover σ from C_6, C_7. If no such tuples exist, $\mathcal{C}$ outputs $\perp$. Otherwise, $\mathcal{C}$ returns m to $\mathcal{A}$.

(e) $\mathcal{O}_{decr}(ID_i, \overline{ID}_i, S, \overline{S}, c, RT, C_R)$: $\mathcal{C}$ recovers the plaintext m as follows:
- If $(((C_R, ID_i, S, \overline{S}, c, RT, 1, 0, 0) \in RET \vee (C_R, ID_i, S, \overline{S}, c, RT, 0, 0, 1) \in RET) \wedge (\overline{ID}_{i'}, sk_{\overline{ID}_{i'}}) \in SKT)$, $\mathcal{C}$ recovers m using $sk_{\overline{ID}_{i'}}$, where $\overline{ID}_{i'} \in \overline{S}$.
- If $((C_R, ID_i, S, \overline{S}, c, RT, 1, 0, 0) \in RET \wedge (\overline{ID}_{i'}, sk_{\overline{ID}_{i'}}) \notin SKT)$ or $\mathcal{A}$ generates C_R using the re-encryption key which is generated correctly by itself, $\mathcal{C}$ checks whether there exist two tuples (m, α, k), (θ, α', k') in H_0^{List} and two tuples (R, δ_1), (R', δ_1') in H_2^{List}, such that $C_1 = w^{-k}$, $C_2' = (\frac{R^{\beta^{-1}}}{e(w^{-1}, h^{B_{i,\bar{s}(\gamma)}})^k})^{H_5(\theta)}$, $\bar{C_3} \oplus H_{10}(\sigma) = (m||\alpha) \oplus \delta_1$, $C_4 = H_3(c, S, RT)^k$, $rk_1 = w^{-k'}$, $rk_2 = h^{k' \cdot \prod_{i'=1}^{s'} (\gamma + H_1(\overline{ID}_{i'}))}$, $rk_3 = (\theta||\alpha') \oplus \delta_1'$, $rk_4 = H_6(RT, c, rk_1, rk_2, rk_3)^{k'}$, $R = e(g', h)^k$ and $R' = e(g', h)^{k'}$, where $\bar{s} = |S|, s' = |\overline{S}|$. Recall that $\mathcal{C}$ can compute τ, and recover σ from C_6, C_7. If no such tuples exist, $\mathcal{C}$ outputs $\perp$. Otherwise, $\mathcal{C}$ sends m to $\mathcal{A}$.

- Otherwise, $\mathcal{C}$ recovers m as above paragraph except that $\mathcal{C}$ needs to recover the random re-encryption key $rk_{ID_i \to \overline{S}|RT,c}$ from RKT, and verifies whether C_2' is correctly constructed from rk_0, rk_5, C_2 and C_4. Despite C_2 is not included in C_R, it can be reconstructed by $\mathcal{C}$ with knowledge of k.

4. **Challenge.** When $\mathcal{A}$ decides that Phase I is over, it outputs m_0, m_1, c^*, RT^* to $\mathcal{C}$. $\mathcal{C}$ responds as follows.
   (a) Compute $C_1^* = g_0^{-k\cdot\gamma\cdot f(\gamma)}$ and $C_2^* = h_0^{k\cdot g(\gamma)}$.
   (b) Flip a random coin $b \in \{0,1\}$, choose $\alpha^* \in_R \{0,1\}^\lambda$, $\sigma^* \in_R \{0,1\}^{2\lambda}$, $\bar{C_3}^* \in_R \{0,1\}^{2\lambda}$, implicitly define $C_3^* = \bar{C_3^*} \oplus H_{10}(\sigma^*)$, $H_2(T^{\prod_{i=t+s+1}^{t+n} x_i} \cdot e(g_0^{k\cdot\gamma\cdot f(\gamma)}, h_0^{q(\gamma)})) = C_3^* \oplus (m_b||\alpha^*)$, where $q(\gamma) = \frac{1}{\gamma}\cdot(\prod_{i=t+s+1}^{t+n}(\gamma + x_i) - \prod_{i=t+s+1}^{t+n} x_i)$ and $x_i = H_1(ID_i)$.
   (c) Compute $\bar{k}^* = H_9(\sigma^*, C_3^*)$, $C_6^* = g^{\bar{k}^*}$ and $C_7^* = \sigma^* \oplus H_8(\bar{e}(H_7(RT^*), TP)^{\bar{k}^*})$.
   (d) Issue an H_3 query on (c^*, S^*, RT^*) to achieve the tuple (c^*, RT^*, S^*, δ_2^*, ξ_1^*), and define $C_4^* = (g_0^{k\cdot\gamma\cdot f(\gamma)})^{\delta_2^*}$.
   (e) Issue an H_4 query on (C_1^*, $\bar{C_3^*}$, C_4^*, C_6^*, C_7^*) to achieve the tuple (C_1^*, $\bar{C_3^*}$, C_4^*, C_6^*, C_7^*, δ_3^*, ξ_2^*), and define $C_5^* = (g_0^{k\cdot\gamma\cdot f(\gamma)})^{\delta_3^*}$.
   (f) Output the challenge ciphertext $C^* = (RT^*, C_1^*, C_2^*, \bar{C_3^*}, C_4^*, C_5^*, C_6^*, C_7^*)$.

   If $T = e(g_0, h_0)^{k\cdot f(\gamma)}$, C^* is a valid ciphertext. Implicitly letting $H_0(m_b, \alpha^*) = k$, one can verify that

$$
\begin{aligned}
&C_1^* = g_0^{-k\cdot\gamma\cdot f(\gamma)} = w^{-k}, C_2^* = h_0^{k\cdot g(\gamma)} = h_0^{k\cdot \prod_{i=t+s+1}^{t+n}(\gamma+x_i)\cdot \prod_{i=t+1}^{t+s}(\gamma+x_i)} \\
&= h^{k\cdot \prod_{i=t+1}^{t+s}(\gamma+H_1(ID_i^*))}, \\
&\bar{C_3^*} = \bar{C_3^*} \oplus (H_{10}(\sigma^*) \oplus (m_b||\alpha^*)) \oplus (H_{10}(\sigma^*) \oplus (m_b||\alpha^*)) \\
&= H_2(T^{\prod_{i=t+s+1}^{t+n} x_i} \cdot e(g_0^{k\cdot\gamma\cdot f(\gamma)}, h_0^{q(\gamma)})) \oplus (m_b||\alpha^*) \oplus H_{10}(\sigma^*) \\
&= (m_b||\alpha^*) \oplus H_2(e(g', h)^k) \oplus H_{10}(\sigma^*), \\
&C_4^* = (g_0^{k\cdot\gamma\cdot f(\gamma)})^{\delta_2^*} = (g_0^{\gamma\cdot f(\gamma)\cdot\delta_2^*})^k = H_3(c^*, S^*, RT^*)^k, \\
&C_5^* = (g_0^{k\cdot\gamma\cdot f(\gamma)})^{\delta_3^*} = (g_0^{\gamma\cdot f(\gamma)\cdot\delta_3^*})^k = H_4(C_1^*, \bar{C_3^*}, C_4^*, C_6^*, C_7^*)^k.
\end{aligned}
$$

   However, if $T = T' \in_R \mathbb{G}_T$, the challenge ciphertext is independent of the bit b in the view of $\mathcal{A}$.

5. **Phase II.** $\mathcal{A}$ continues issuing queries as in Phase I with the constraints defined in Definition 2.
6. **Guess.** $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$. If $b' = b$, $\mathcal{C}$ outputs 1 (i.e. deciding $T = e(g_0, h_0)^{k\cdot f(\gamma)}$); otherwise, $\mathcal{C}$ outputs 0 (i.e. deciding $T = T'$).

The advantage of $\mathcal{C}$ against the (P, Q, f)-$GDDHE$ assumption is at least

$$
\epsilon' \geq \frac{1}{q_{H_2}}(\epsilon - \frac{q_{H_0} + (q_{H_0} + q_{H_2})(q_{d_1} + q_{d_2})}{2^{2\lambda}} - \frac{2(q_{d_1} + q_{d_2}) + q_{re}}{q}),
$$

and the running time of $\mathcal{C}$ is bounded by

$$t' \leq \bar{t} + O(1)(q_{H_i} + q_{sk} + q_{rk} + q_{re} + q_{d2} + q_{d1}) + t_e(q_{sk} + q_{H_0}(2q_{re} + 2q_{d2} + 8q_{d1}) + (2n+9)q_{rk} + (3n+8)q_{re} + (n+2)q_{d2} + (2n+4)q_{d1}) + t_p(7q_{re} + 7q_{d2} + 2q_{d1}).$$

This completes the proof of Theorem 1. □

B Proof of Theorem 2

Proof. We prove our scheme in the static notion of Definition 2. Suppose the BF IBE scheme [4] is CCA secure under the BDH assumption. We use the challenger $\mathcal{C}'$ of the BF IBE scheme in a black box manner. Here we separate the ciphertext into two parts: one is the CPBRE part (i.e. C_1, C_2, C_3, C_4, C_5), and the other is the TRE part (i.e. $RT, \bar{C}_3, C_6, C_7$). $\mathcal{A}$ outputs the challenge to $\mathcal{C}$ before setup phase. $\mathcal{C}$ is able to generate public parameters and secret keys (of CPBRE), and response the queries to $\mathcal{O}_{sk}$, $\mathcal{O}_{rk}$ and $\mathcal{O}_{re}$ correctly[11]. When $\mathcal{A}$ issues a query to $\mathcal{O}_{dec}$ or $\mathcal{O}_{decr}$, $\mathcal{C}$ first decrypts the ciphertext of CPBRE part using the corresponding secret key, next forwards the result to $\mathcal{C}'$ and receives σ which can be used to recover the plaintext. For the queries of timed-release keys to $\mathcal{O}_{ts}$, $\mathcal{C}$ can response correctly with the help of $\mathcal{C}'$. In the challenge phase, $\mathcal{C}$ first generates $((m_0||\alpha^*) \oplus H_2(e(g', h)^{k^*}), (m_1||\alpha^*) \oplus H_2(e(g', h)^{k^*}))$ as the challenge messages for $\mathcal{C}'$. After receiving the challenge ciphertext of BF IBE scheme output by $\mathcal{C}'$, $\mathcal{C}$ adds the rest CPBRE part to the ciphertext, and outputs the resulting ciphertext to $\mathcal{A}$. Finally, $\mathcal{C}$ outputs whatever $\mathcal{A}$ outputs.

This completes the proof of Theorem 2. □

[11] Note that $\mathcal{C}$ is allowed to generate all valid secret keys such that it can output all valid re-encryption keys and the corresponding re-encryption.