General Certificateless Encryption and Timed-Release Encryption

Sherman S.M. Chow[1⋆], Volker Roth[2], and Eleanor G. Rieffel[2]

[1] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu
[2] FX Palo Alto Laboratory
3400 Hillview Avenue
Palo Alto, CA 94304, USA
{vroth, rieffel}@fxpal.com

Abstract. While recent timed-release encryption (TRE) schemes are implicitly supported by a certificateless encryption (CLE) mechanism, the security models of CLE and TRE differ and there is no generic transformation from a CLE to a TRE. This paper gives a generalized model for CLE that fulfills the requirements of TRE. This model is secure against adversaries with adaptive trapdoor extraction capabilities, decryption capabilities for arbitrary public keys, and partial decryption capabilities. It also supports hierarchical identifiers. We propose a concrete scheme under our generalized model and prove it secure without random oracles, yielding the first strongly-secure security-mediated CLE and the first TRE in the standard model. In addition, our technique of partial decryption is different from the previous approach.

Key words: security-mediated certificateless encryption, timed-release encryption, standard model

1 Introduction

In identity-based encryption (IBE) [30], encryption is done with respect to any arbitrary string viewed an identifier (ID). Since the birth of practical IBE constructions, this idea has been used to achieve other security goals, such as certificateless encryption (CLE) [1, 15, 17] and timed-release encryption (TRE) [3].

CLE is intermediate between IBE and traditional public key encryption (PKE). Traditional PKE requires a certification infrastructure but allows users to create their own public/private key pairs so that their private keys are truly private. Conversely, IBE avoids the need for certificates at the expense of adding a trusted key generation center (KGC) that generates the private keys, which means the KGC has the capability to decrypt all messages. CLE combines the

⋆ This research is done while the first author was a research intern of FXPAL. We thank Wolfgang Polak for helpful discussions and the reviewers for their feedback.

advantages of both: no certificates are needed and messages can only be decrypted by the recipient. Generally, CLE is constructed by combining IBE and PKE. The existence of the PKE component means that the KGC cannot decrypt messages. Instantaneous revocation is difficult for typical CLE schemes. Security-mediated certificateless encryption (SMCLE) addresses this issue.

In TRE, a message is encrypted under a public key and a time; both the private key and a time-dependent trapdoor are needed for decryption. A time-server is trusted to keep a trapdoor confidential until an appointed time. Apart from delayed release of information, TRE supports many other applications due to its small trapdoor size and its commitment provision (see [13, 19]).

1.1 The difficulty of converting between CLE and TRE

A practical TRE requires system parameters to be small relative to the number of supported time periods. IBE supports an efficient time-based unlock mechanism by treating the identities as time periods [4, 27]. This approach supports only universal disclosure of encrypted documents since one trapdoor can decrypt all ciphertexts for a specific time; the inherent key-escrow property of IBE prohibits the encryption for a designated receiver.

Since CLE is an "escrow-free version" of IBE, and both TRE and CLE are a kind of double-encryption, it is natural to think CLE is what we are looking for to realize a TRE. While most recent TRE schemes can be viewed as containing an implicit CLE mechanism, a generic transformation from CLE to TRE is unlikely to be provable secure [7]. Difficulty in reducing the confidentiality of TRE to that of CLE arises when the adversary is a "curious" time-server. In CLE, an identity is associated with only one public key, so a curious KGC is not allowed to replace the public key associated with an identifier arbitrarily (otherwise, decryption is trivial since it holds both parts of secrets). On the other hand, in TRE a time identifier is never bound to any public key, so the public key associated with a time identifier can be replaced. There is no way to simulate this implicit public key replacement when CLE is viewed as a black box.

There is another subtle difference in modeling of curious users. In a secure multi-user system, the security of a user is preserved even if other users are compromised. In CLE, the user secret key together with the trapdoor given by the KGC give the *full* private key. With the assumption that the user secret key will be securely deleted after the combination, most CLE models assume the adversary can get only trapdoors and full private keys. For most CLE schemes under this model (e.g. [18]), the user secret key cannot be recovered from the trapdoor and the full private key. Moreover, some CLE formulations [2, 25, 31] do not have user secret keys at all. In TRE, user secret keys are held by each user, which makes it impossible to reduce the security of TRE to that of CLE.

1.2 Our Contributions

Our generalized model for CLE overcomes the aforementioned difficulties and has sufficient power to fulfill the requirements of TRE. Our model is secure

against an adversary with adaptive trapdoor extraction capabilities for arbitrary identifiers (instead of selective identifiers, e.g. [4, 28]), decryption capabilities for arbitrary public keys (as considered in strongly-secure CLE [18]) and partial decryption capabilities (as considered in security-mediated CLE [14]). Our model also supports hierarchical identifiers which have not been considered formally for CLE and TRE. Design choices behind our formulation are justified.

All previous concrete TRE schemes [3, 7–9, 12, 16, 19, 22, 24], and the only concrete SMCLE scheme [14], were proven in the random oracle model. Our model is strong but achievable: our proposed scheme is the first strongly-secure SMCLE. With our security-preserving transformation from a general CLE to a TRE, it also yields the first TRE in the standard model.

This work enriches the study of SMCLE by providing a novel partial decryption technique which is different from that in [14], and enriches TRE by supporting a new business model for the time-server. Finally, hierarchy of identifiers makes decryption of ciphertext for passed periods more manageable.

2 Related Work

2.1 Timed-Release Encryption

Early TRE schemes require interaction with the time-server. Rivest, Shamir and Wagner's idea [29] require senders to reveal the release-time of the messages in their interactions with the server, so the senders cannot be anonymous to the server. In Di Crescenzo, Ostrovsky and Rajaopalan 's scheme [16], it is the receiver who interacts with the time-server by invoking a "conditional oblivious transfer protocol", which is computationally intensive.

Blake and Chan made the first attempt to construct a non-interactive TRE [3]. The formal security model of message confidentiality was later considered independently by Cheon *et al.* [12] and Cathalo, Libert and Quisquater [7]. The former focuses on authenticated TRE. The latter also formalizes the release-time confidentiality. The recovery of past time-dependent trapdoors from a current trapdoor was studied in [9] and [27], which employs a hash chain and a tree structure [6] respectively. The study of the pre-open capability in TRE was initiated in [24] and improved by [19]. Recently, Chalkias, Hristu-Varsakelis and Stephanides proposed an efficient TRE scheme [8] with random oracles.

2.2 Certificateless Encryption

Al-Riyami and Paterson [1] proposed certificateless encryption in 2003. Extensive surveys of CLE security models and constructions can be found in [15, 17]. Two types of adversaries are considered in certificateless encryption. A Type-I adversary models coalitions of rogue users without the master secret. Due to the lack of a certificate, the adversary is allowed to replace the public keys of users. A Type-II adversary models a curious KGC who has the master key but cannot replace the public keys of any users. In Al-Riyami and Paterson's security model

for encryption [1], a Type-I adversary can ask for the decryption of a ciphertext under a replaced public key. Schemes secure against such attacks are called "strongly-secure" [18], and the oracle is termed a "strong decryption oracle". A weaker type of adversary, termed Type-I$^-$, can only obtain a correct plaintext if the ciphertext is submitted along with the corresponding user secret key.

The Al-Riyami and Paterson scheme [1] is secure against both Type-I and Type-II adversaries in the random oracle model. It was believed [25, 26, 28] that [26] gave the first CLE in the standard model. However, it is possible to instantiate a prior generic construction in [14] with a PKE and an IBE in the standard model to obtain a secure CLE without random oracles. Both [26] and the instantiation of [14] are only secure against Type-I$^-$ attacks. Based on [20], a selective-ID secure CLE without random oracles was proposed [28]. This scheme cannot be efficiently extended to a TRE since the user's public key is dependent on the identity, which is never coupled with a fixed time-identifier in TRE. Recently, the first strongly-secure CLE in the standard model is proposed [18].

Al-Riyami and Paterson give an extension for hierarchical CLE [1]. However, no security model is given. We are not aware of any literature with formal work on hierarchical CLE, particularly none proven secure in the standard model.

Baek et al. proposed the first CLE that does not use pairings [2]. The CLE proposal [25] uses similar ideas, but their security proof ignores the public key replacement of the target user being attacked. This limitation is removed in Sun, Zhang and Baek's work [31]. To replace the pairing, these schemes make part of the user's public key dependent on the identity-specific trapdoor given by the KGC, which means TRE cannot be obtained trivially from these constructions.

Security-mediated certificateless encryption (SMCLE), introduced by Chow, Boyd and González Nieto [14], adds a security-mediator (SEM) who performs partial decryption for the user by request. This idea gives a more general treatment of the decryption queries in the CLE paradigm: the adversary can ask for partial decryption results under either the SEM trapdoor generated by the KGC or the user secret key A concrete construction in the random oracle model and a generic construction in the standard model are proposed in [14]. Prior to our work, no strongly-secure SMCLE existed in the standard model.

3 General Security-Mediated Certificateless Encryption

3.1 Notation

We use an ID-vector $\overrightarrow{ID} = (\mathsf{ID}_1, \mathsf{ID}_2, \cdots, \mathsf{ID}_L)$ to denote a hierarchy of identifiers $(\mathsf{ID}_1, \mathsf{ID}_2, \cdots, \mathsf{ID}_L)$. The length of $\overrightarrow{ID}$ is denoted by $|\overrightarrow{ID}| = L$. Let $\overrightarrow{ID}||\mathsf{ID}_r$ denote the vector $(\mathsf{ID}_1, \mathsf{ID}_2, \cdots, \mathsf{ID}_L, \mathsf{ID}_r)$ of length $|\overrightarrow{ID}| + 1$. We say that $\overrightarrow{ID}$ is a prefix of $\overrightarrow{ID'}$ if $|\overrightarrow{ID}| \leq |\overrightarrow{ID'}|$ and $\mathsf{ID}_i = \mathsf{ID}'_i$ for all $1 \leq i \leq |\overrightarrow{ID}|$. We use $\emptyset$ to denote an empty ID-vector where $|\emptyset| = 0$ and $\emptyset||\mathsf{ID}_r = \mathsf{ID}_r$. Finally, we use the notation $(\{0,1\}^n)^{\leq h}$ to denote the set of vectors of length less than or equal to h, where each component is a n-bit long bit-string.

3.2 Syntax

We propose a new definition of the (security-mediated) certificateless encryption, which also extends the definition of a 1-level SMCLE scheme in [14] to h levels.

Definition 1. *An h-level SMCLE scheme for identifiers of length n is defined by the following sextuple of* PPT *algorithms:*

- Setup *(run by the server) is a probabilistic algorithm which takes a security parameter 1^λ, outputs a master secret key Msk (which can also be denoted as $d_\emptyset$), and the global parameters Pub (which include $h = h(\lambda)$ and $n = n(\lambda)$ implicitly) We assume all other algorithms take Pub implicitly as an input.*
- $\mathsf{Extract}$ *(run by the server or any one who holds a trapdoor) is a possibly probabilistic algorithm which takes a trapdoor $d_{\overrightarrow{ID}}$ corresponding to an h-level identifier $\overrightarrow{ID} \in (\{0,1\}^n)^{\leq h}$, and a string $\mathsf{ID}_r \in \{0,1\}^n$, outputs a trapdoor key $d_{\overrightarrow{ID}||\mathsf{ID}_r}$ associated with the ID-vector $\overrightarrow{ID}||\mathsf{ID}_r$. The master secret key Msk is a trapdoor corresponding to a 0-level identifier.*
- KGen *(run by a user) is a probabilistic algorithm which generates a public/private key pair $(\mathsf{pk}_u, \mathsf{sk}_u)$.*
- Enc *(run by a sender) is a probabilistic algorithm which takes a message m from some implicit message space, an identifier $\overrightarrow{ID} \in (\{0,1\}^n)^{\leq h}$, and the receiver's public key pk_u as input , returns a ciphertext C.*
- Dec^S *(run by any one who holds the trapdoor, either a SEM in SMCLE or a receiver in CLE) is a possibly probabilistic algorithm which takes a ciphertext C and a trapdoor key $d_{\overrightarrow{ID}}$, returns either a token D which can be seen as a partial decryption, or an invalid flag $\perp$ (which is not in the message space).*
- Dec^U *(run by a receiver) is a possibly probabilistic algorithm which takes the ciphertext C, the receiver's secret key sk_u and a token D as input, returns either the plaintext, an invalid flag $\perp_D$ denoting D is an invalid token, or an invalid flag $\perp_C$ denoting the ciphertext is invalid.*

For correctness, we require that $\mathsf{Dec}^U(C, \mathsf{sk}, \mathsf{Dec}^S(C, \mathsf{Extract}(\mathsf{Msk}, \overrightarrow{ID}))) = m$ *for all $\lambda \in \mathbb{N}$, all* $(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, *all* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}$, *all message m, all ID-vector $\overrightarrow{ID}$ in $(\{0,1\}^n)^{\leq h}$ and all* $C \xleftarrow{\$} \mathsf{Enc}(m, \overrightarrow{ID}, \mathsf{pk})$.

3.3 Security

Each adversary has access to the following oracles:

1. An $\mathsf{ExtractO}$ oracle that takes an ID-vector $\overrightarrow{ID} \in (\{0,1\}^n)^{\leq h}$ as input and returns its trapdoor $d_{\overrightarrow{ID}}$.
2. An UskO oracle that takes a public key pk as input and returns its corresponding private key sk.
3. A DecO^S oracle that takes a ciphertext C and an ID-vector $\overrightarrow{ID}$, and outputs $\mathsf{Dec}^S(C, d_{\overrightarrow{ID}})$. Note that C may or may not be encrypted under $\overrightarrow{ID}$.
4. A DecO^U oracle that takes a ciphertext C, a public key pk and a token D, and outputs $\mathsf{Dec}^U(C, \mathsf{sk}, D)$ where sk is the secret key that matches pk.

5. A DecO oracle that takes a ciphertext C, an ID-vector $\overrightarrow{ID}$, and a public key pk; outputs $\mathsf{Dec}^U(C, \mathsf{sk}, D)$ where sk is the secret key that matches pk, $D = \mathsf{Dec}^S(C, d_{\overrightarrow{ID}})$ and C may or may not be encrypted under $\overrightarrow{ID}$ and pk.

Following common practice, we consider the two kinds of adversaries.

1. A Type-I adversary that models any coalition of rogue users, and who aims to break the confidentiality of another user's ciphertext.
2. A Type-II adversary that models a curious KGC, who aims to break the confidentiality of a user's ciphertext[3].

We use the common security model in which the adversary plays a two-phased game against a challenger. The game is modeled by the experiment below, $\mathsf{X} \in \{\mathrm{I}, \mathrm{II}\}$ denotes whether an PPT adversary $\mathcal{A} = (\mathcal{A}_{\mathsf{find}}, \mathcal{A}_{\mathsf{guess}})$ is of Type-I or II, and determines the allowed oracle queries $\mathcal{O}$ and the auxiliary data Aux.

Definition 2. Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-X}}(\lambda)$

$(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$(m_0, m_1, \mathsf{pk}^*, \overrightarrow{ID}^*, \mathsf{state}) \xleftarrow{\$} \mathcal{A}_{\mathsf{find}}^{\mathcal{O}}(\mathsf{Pub}, \mathsf{Aux})$
$b \xleftarrow{\$} \{0,1\}$, $C^* \xleftarrow{\$} \mathsf{Enc}(m_b, \overrightarrow{ID}^*, \mathsf{pk}^*)$
$b' \xleftarrow{\$} \mathcal{A}_{\mathsf{guess}}^{\mathcal{O}}(C^*, \mathsf{state})$
If $(|m_0| \neq |m_1|) \vee (b \neq b')$ *then return 0 else return 1*

$\mathcal{O}$ *is a set of oracles* $\mathsf{ExtractO}(\cdot), \mathsf{UskO}(\cdot), \mathsf{DecO}^S(\cdot,\cdot), \mathsf{DecO}^U(\cdot,\cdot,\cdot), \mathsf{DecO}(\cdot,\cdot,\cdot)$.

Variables marked with * refer to challenges by the adversary. The adversary chooses a public key pk^* and an ID-vector $\overrightarrow{ID}^*$ to be challenged with, and the challenger returns a challenge ciphertext C^*. The following two definitions prohibit the adversary from trivially using the oracles to query for the answer to (parts of) the challenge.

Definition 3. *A hierarchical security-mediated certificateless encryption scheme is* (t, q_E, q_D, ϵ) *CCA-secure against a Type-I adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-I}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_E *extraction queries and* q_D *decryption queries (of any type), subjects to the following constraints:*

1. $\mathsf{Aux} = \emptyset$*, i.e. no auxiliary information is given to the adversary.*
2. *No* $\mathsf{ExtractO}(\overrightarrow{ID}')$ *query throughout the game, where* $\overrightarrow{ID}'$ *is a prefix of* $\overrightarrow{ID}^*$.
3. *No* $\mathsf{UskO}(\mathsf{pk})$ *query throughout the game for any* pk.
4. *No* $\mathsf{DecO}^S(C^*, \overrightarrow{ID}^*)$ *query throughout the game.*
5. *No* $\mathsf{DecO}(C^*, \overrightarrow{ID}^*, \mathsf{pk}^*)$ *query throughout the game.*

All public keys in the game are chosen by the adversary. It is natural to assume the adversary knows the corresponding secret keys.

[3] A rogue SEM is weaker than a Type-II adversary.

Definition 4. *A hierarchical security-mediated certificateless encryption scheme is* (t, q_K, q_D, ϵ) *CCA-secure against a Type-II adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-II}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_D *decryption queries (of any type), subjects to the following conditions:*

1. $\mathsf{Aux} = (\mathsf{Msk}, \{\mathsf{pk}_1^*, \cdots, \mathsf{pk}_{q_K}^*\})$*, i.e.* $\mathcal{A}$ *is given the master secret and a set of challenge public keys.*
2. $\mathsf{pk}^* \in \{\mathsf{pk}_1^*, \cdots, \mathsf{pk}_{q_K}^*\}$*, i.e. the challenge public key must be among the set given by the challenger.*
3. *No* $\mathsf{UskO}(\mathsf{pk})$ *query throughout the game if* $\mathsf{pk} \notin \{\mathsf{pk}_1^*, \cdots, \mathsf{pk}_{q_K}^*\}$ *or* $\mathsf{pk} = \mathsf{pk}^*$*.*
4. *No* $\mathsf{DecO}^U(C^*, \mathsf{pk}^*, D)$ *query throughout the game, where* D *is outputted by the algorithm* $\mathsf{Dec}^S(C^*, d_{\overrightarrow{ID}^*})$*.*
5. *No* $\mathsf{DecO}(C^*, \overrightarrow{ID}^*, \mathsf{pk}^*)$ *query throughout the game.*

Since Msk is given to the adversary, the challenge public key must be in the set given by the challenger.

3.4 Discussions on Our Choices for Definition

This section explains the intuitions behind the choices made in formulating our definition and highlights the relationship between existing definitions and ours.

User key generation. In order to support more general applications like TRE, the interface for the algorithms needs a more general syntax. A subtle change is that our user key generation algorithm KGen only takes the system parameter as input but *not* the identifier. In some CLE schemes [2, 25, 28, 31] the inclusion of the identifier, or the trapdoor for an identifier, is *essential* for the generation of the user public key. For these schemes, KGen can be executed only after $\mathsf{Extract}$, so straightforward adaption results in inefficient TREs in which the size of the user public key grows linearly with the number of supported time periods.

Simplification of Type-I adversary. In existing models for 1-level CLE [1, 18], $\mathsf{ExtractO}$ query of $\overrightarrow{ID}^*$ is allowed; if such a query is issued, the challenge public key pk^* can no longer be chosen by the adversary. In our discussion, we separate this behavior from the Type-I model and consider this type of adversarial behavior ($\mathsf{ExtractO}(\overrightarrow{ID}')$ where $\overrightarrow{ID}'$ is a prefix of $\overrightarrow{ID}^*$) as a weaker variant of, and hence covered by, a Type-II adversary. It is true that our resulting definition for Type-I adversary is weaker, but the "missing part" is not omitted from the security requirement since CLEs must consider Type-II adversaries; this simplification was justified and adopted in [23, Section 2.3].

Existing models also allow full private key extraction for the public keys prepared by the challenger. In our Type-I game, all of the public keys to be attacked are generated by the adversary, so UskO query is prohibited. The remaining scenario, where the adversary intends to attack a public key given by the challenger, is also a weaker variant of our Type-II model. To conclude, we keep the essence of the existing models, and include the adversarially chosen public keys (for Type-I) and UskO (for Type-II) to match with TRE.

Strong decryption oracle. In our definition, the decryption oracle works even if the public key is adversarially chosen but the secret key is not supplied. The original definition of CLE [1] does not allow a strong decryption oracle for curious KGC adversary, but it is considered in recent work [18]. Adding the following restriction weakens Definition 4 to correspond to a Type-II$^-$ attack:

5. *(Type-II$^-$) No* $\mathsf{DecO}(C, \overrightarrow{ID}, \mathsf{pk})$ *query throughout the game for any* C *if* $\mathsf{pk} \notin \{\mathsf{pk}_1^*, \cdots \mathsf{pk}_{q_K}^*\}$*, unless the corresponding secret key* sk *is supplied when the* DecO *query is made.*

The Type-I$^-$ game can be obtained by adding $\mathsf{Aux} = \{\mathsf{pk}_1^*, \cdots \mathsf{pk}_{q_K}^*\}$ and the above restriction to Definition 3, but with a restriction on UskO as in Definition 4.

Implicit public key replacement. In our generalization of CLE, we "remove" (i.e. make implicit) the oracle for replacing the public key corresponding to an identifier. This change may affect the following choices:

1. The adversary's choice of the victim user it wishes to be challenged with,
2. The choice of user in decryption oracle queries.

However, there are other "interfaces" in our model such that the adversary can still make the above choices. Our model still allows the adversary to choose which identifier/public key it wants to attack. For decryption queries, the adversary can just supply different combination of identifier and public key to the DecO^S and DecO^U oracles. In this way, implicit replacement is done. In other words, when compared with the original model [1], the security model is not weakened, but generalized to cover applications of CLE such as TRE.

Reason for "removing" public key request and replacement oracles. In traditional definitions of CLE [1], oracles for retrieving and replacing public key depend upon the fact that an identifier is always bound to a particular user. Replacing a user's public key means changing the public key associated with a certain identifier. In TRE, identifiers correspond to policies governing the decryption, so a single identifier may be "shared" among multiple users. For this reason, our model must be free from the concept of "user = identifier".

Alternative definition of public key replacement. What about allowing a restricted public key replacement, such that a public key associated with an identifier can be replaced by a public key associated with another identifier, but not an arbitrary one supplied by the adversary? This definition still requires an identifier to belong to a single user. Moreover, this definition makes the treatment of a strong decryption oracle complicated: the idea of restricted replacement among a fixed set of public keys does not match well with decrypting under adversarially chosen public keys.

SMCLE is more general than plain CLE. The two separate decryption oracles in the SMCLE model provide a more general notion than CLE:

1. Some CLE schemes are not CCA-secure when the adversary has access to a partial decryption oracle (see [14]).

2. Since the decryption oracle is separated in two, the SMCLE model does not have the notion of a "full" private key which is present in previous CLE models (a full private key is a single secret for the complete decryption of the ciphertext). On the ground that separated secrets can always be concatenated into a single full one, this simplification (of private key) has already been adopted in more recent models [23].

Difference with the previous SMCLE definition. Our user decryption oracle DecO^U returns different invalid flags for the cases of invalid token from the SEM or invalid ciphertext. This distinction was not captured in [14].

User decryption oracle in SMCLE. To exclude trivial attacks, our Type-II adversary model disallows the challenge ciphertext C^* to be decrypted by the decryption oracle under the challenge public key and a token D obtained from the algorithm (not the oracle) $\mathsf{Dec}^S(C^*, \mathsf{ID}^*)$, where ID^* is the challenge identifier. To implement this restriction, our new SMCLE definition checks whether a token D is a *valid* token, corresponding to a ciphertext and an identifier.

While our security definition is tightly coupled with the ability to check the token, we think that it is natural for the user to be able to perform such a test (especially if the user pays for each token). Even without an explicit testing algorithm, the challenger may do the test as well since it simulates the scheme's execution. It gives a weaker definition if we prohibit any decryption query for the challenge ciphertext under the challenge public key, irrespective of the token.

Justifications for our definition of hierarchical CLE. In the hierarchical scheme of [1], an entity at level k derives a trapdoor for its children at level $k+1$ using both its trapdoor and its secret key. In our proposed model, a level k entity uses only the trapdoor obtained from its parent at level $k-1$ to derive keys for its children. We do not see any practical reason for requiring the secret key in the trapdoor derivation. Our definition avoids certain complications: for example, in [1], the decryption requires the public keys of all the ancestors.

We do allow the decryption of the ciphertext under $\overrightarrow{ID}'$ which is a prefix of $\overrightarrow{ID}^*$. This is stronger than the counterpart in some hierarchical IBE models [21].

Theorem 1 *If there exists a secure* 1*-level SMCLE scheme under Definition 3 and 4, there exists a CLE scheme which is secure under the definition of [1].*

Proof. Our aim is to build a simulator $\mathcal{B}$ which uses an adversary $\mathcal{A}$ of CLE to break the security of our 1-level SMCLE scheme. The simulator basically forwards everything (the system parameters, the oracle queries and responses, and the guess) back and forth between its own SMCLE challenger and the CLE adversary. Faced with a Type-II adversary of CLE, the simulator acts as a Type-II security of 1-level SMCLE. For a Type-I adversary of CLE, $\mathcal{B}$ flips a fair coin to determine its guess whether $\mathcal{A}$ will issue an $\mathsf{ExtractO}$ query of $\overrightarrow{ID}^*$. If it guesses not, $\mathcal{B}$ just plays the Type-I game as usual. If it guesses so, $\mathcal{B}$ will try to use $\mathcal{A}$ to win the Type-II game of SMCLE instead. The $\mathsf{ExtractO}$ query can be answered by $\mathcal{B}$ because it owns Msk now. The reduction tightness is reduced by a factor of 2. This simple trick is also used in [18, Appendix B, Game 4].

We omit the details for most queries, but focus on the distinctions that involve public key requests and replacement. The simulator must maintain a table to store the binding between an identifier and a public key. Whenever a Type-I adversary issues a public key request query, $\mathcal{B}$ executes $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}$, stores sk (so $\mathcal{B}$ can reply if $\mathcal{A}$ asks for it), and returns pk. For a Type-II adversary, $\mathcal{B}$ picks a random public key from $\{\mathsf{pk}_1^*, \cdots, \mathsf{pk}_{q_K}^*\}$ and assigns it as the public key of the queried ID. When $\mathcal{A}$ makes a key replacement query, the simulator updates its own table. For every other request regarding a particular identifier, the simulator retrieves the corresponding public key from its table and queries its own challenger accordingly. Finally, decryption queries of the CLE adversary are answered by combining results from the two partial decryption oracles. □

4 Our Proposed Construction

4.1 Preliminaries

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative groups of prime order p for which there exists an efficiently computable bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that

1. *Bilinearity*: For all $u, v \in \mathbb{G}$ and $r, s \in \mathbb{Z}_p$, $\hat{e}(u^r, v^s) = \hat{e}(u, v)^{rs}$.
2. *Non-degeneracy*: $\hat{e}(u, v) \neq 1_{\mathbb{G}_T}$ for all $u, v \in \mathbb{G} \setminus \{1_{\mathbb{G}}\}$.

Our scheme's security relies on the intractability of the following problems:

Definition 5. *The* Decision **3**-Party Diffie-Hellman Problem *(3-DDH) in* $\mathbb{G}$ *is to decide if* $T = g^{\beta\gamma\delta}$ *given* $(g, g^\beta, g^\gamma, g^\delta, T) \in \mathbb{G}^5$. *Formally, defining the advantage of a* PPT *algorithm* $\mathcal{D}$, $Adv_{\mathcal{D}}^{3-\mathrm{DDH}}(\lambda)$, *as*

$$\begin{aligned} &| \Pr[1 \xleftarrow{\$} \mathcal{D}(g, g^\beta, g^\gamma, g^\delta, T) | T \leftarrow g^{\beta\gamma\delta} \wedge \beta, \gamma, \delta \xleftarrow{\$} \mathbb{Z}_p^*] \\ &- \Pr[1 \xleftarrow{\$} \mathcal{D}(g, g^\beta, g^\gamma, g^\delta, T) | T \xleftarrow{\$} \mathbb{G} \wedge \beta, \gamma, \delta \xleftarrow{\$} \mathbb{Z}_p^*]|. \end{aligned}$$

We say 3-DDH is intractable if $Adv_{\mathcal{D}}^{3-\mathrm{DDH}}(\lambda)$ *is negligible in* λ *for all* PPT $\mathcal{D}$.

Compared with the Bilinear Diffie-Hellman (BDH) problem, the problem instance of 3-DDH is purely in $\mathbb{G}$ while that of BDH contains an element $\hat{t} \in \mathbb{G}_T$. If BDH problem is solvable, one can solve 3-DDH by feeding $(g, g^\beta, g^\gamma, g^\delta, \hat{e}(g, T))$ to a BDH oracle. The above assumption has been employed in [18].

We introduce a variant of the weak Bilinear Diffie-Hellman Inversion (wBDHI) assumption [4] below in the favor of 3-DDH. The original h-wBDHI problem in $(\mathbb{G}, \mathbb{G}_T)$ [4] is to decide whether $\hat{t} = \hat{e}(g, g^\gamma)^{\alpha^{h+1}}$. The term "inversion" comes from the equivalence to the problem of deciding whether $\hat{t} = \hat{e}(g, g^\gamma)^{1/\alpha}$.

Definition 6. *The* h-Weak Diffie-Hellman Inversion Problem *(h-wDHI) in* $\mathbb{G}$ *is to decide if* $T = g^{\gamma\alpha^{h+1}}$ *given* $(g, g^\gamma, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^h}, T) \in \mathbb{G}^{h+3}$. *Formally, defining the advantage of a* PPT *algorithm* $\mathcal{D}$ *as*

$$\begin{aligned} Adv_{\mathcal{D}}^{h-\mathrm{wDHI}}(\lambda) = &| \Pr[1 \xleftarrow{\$} \mathcal{D}(g, g^\gamma, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^h}, T) | T \leftarrow g^{\gamma\alpha^{h+1}} \wedge \alpha, \gamma \xleftarrow{\$} \mathbb{Z}_p^*] \\ &- \Pr[1 \xleftarrow{\$} \mathcal{D}(g, g^\gamma, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^h}, T) | T \xleftarrow{\$} \mathbb{G} \wedge \alpha, \gamma \xleftarrow{\$} \mathbb{Z}_p^*]|. \end{aligned}$$

We say h-wDHI is intractable if $Adv_{\mathcal{D}}^{h-\mathrm{wDHI}}(\lambda)$ *is negligible in* λ *for all* PPT $\mathcal{D}$.

We require a family of collision resistant hash functions $\mathcal{H}$ too.

Definition 7. *A hash function* $H \xleftarrow{\$} \mathcal{H}(\lambda)$ *is* collision resistant *if*

$$Adv_{\mathcal{C}}^{\mathrm{CR}}(\lambda) = \Pr[H(x) = H(y) \wedge x \neq y | (x, y) \xleftarrow{\$} \mathcal{C}(1^{\lambda}, H) \wedge H \xleftarrow{\$} \mathcal{H}(\lambda)]$$

is negligible as a function of the security parameter λ *for all* PPT *algorithms* $\mathcal{C}$.

4.2 Proposed Construction

Our construction is an h-level generalization of the concrete construction for 1-level in [18]. While [18] uses the technique of [5] to achieve strong decryption oracle, we use the same technique for a different purpose, which is a new way (other than the only known way in [14]) to support partial decryption oracle.

$\mathsf{Setup}(1^{\lambda}, n)$: Let $\mathbb{G}$, $\mathbb{G}_T$ be two multiplicative groups with a bilinear map $\hat{e}$ as defined before. They are of the same order p, which is a prime and $2^{\lambda} < p < 2^{\lambda+1}$.

- **Encryption key:** choose two generators $g, g_2 \in_R \mathbb{G}$.
- **Master public key:** choose an exponent $\alpha \in_R \mathbb{Z}_p$ and set $g_1 = g^{\alpha}$.
- **Hash key for identifier-based key derivation:** choose h many $(\ell+1)$-length vectors $\vec{U}_1, \cdots, \vec{U}_h \in_R \mathbb{G}^{\ell+1}$, where each $\vec{U}_j = (u'_j, u_{j,1}, \cdots, u_{j,\ell})$, $1 \leq j \leq h$. ℓ is a tunable parameter which is a factor of n and $1 \leq \ell \leq n$. Each vector $\vec{U}_j$ $(1 \leq j \leq h)$ corresponds to the j-th level of the hierarchy. We use the notation $\overrightarrow{ID} = (\mathsf{ID}_1, \cdots, \mathsf{ID}_j, \cdots, \mathsf{ID}_k)$ to denote a hierarchy of k n-bit string ID_j's. We write ID_j as ℓ blocks each of length n/ℓ bits $(\mathsf{ID}_{j,1}, \cdots, \mathsf{ID}_{j,\ell})$. We define $F_{\vec{U}_j}(\mathsf{ID}_j) = u'_j \prod_{i=1}^{\ell} u_{j,i}^{\mathsf{ID}_{j,i}}$.
- **Hash key for ciphertext validity:** choose an $(n+1)$-length vector $\vec{V} = (v', v_1, \cdots, v_n) \in_R \mathbb{G}^{n+1}$. This vector defines the hash function $F_{\vec{V}}(w) = v' \prod_{j=1}^{n} {v_j}^{b_j}$ where w is a n-bit string $b_1 b_2 \cdots b_n$.
- **Hash function:** pick a function $H : \{0,1\}^* \to \{0,1\}^n$ from a family of collision-resistant hash functions according to the parameter λ.

The public parameters Pub and the master secret key Msk are given by

$$\mathsf{Pub} = (\lambda, p, \mathbb{G}, \mathbb{G}_T, \hat{e}(\cdot,\cdot), n, \ell, g, g_1, g_2, \vec{U}_1, \cdots, \vec{U}_h, \vec{V}, H(\cdot)), \qquad \mathsf{Msk} = g_2^{\alpha}.$$

We require the discrete logarithms (with respect to g) of all $\mathbb{G}$ elements in Pub except g, g_1 to be unknown to the KGC. In practice, these elements can be generated from a pseudorandom function of a public seed.

$\mathsf{Extract}(d_{\overrightarrow{ID}}, \mathsf{ID}_r)$: For $\overrightarrow{ID} = (\mathsf{ID}_1, \cdots, \mathsf{ID}_k)$ for $k \leq h$, a trapdoor is in the form:

$$d_{\overrightarrow{ID}} = (a_1, a_2, \vec{z}_{k+1}, \cdots, \vec{z}_h) = (g_2^{\alpha} \cdot (\prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j))^r, g^r, (\vec{U}_{k+1})^r, \cdots, (\vec{U}_h)^r),$$

where $r \in_R \mathbb{Z}_p^*$ and $(\vec{U}_j)^r = ((u'_j)^r, (u_{j,1})^r, \cdots, (u_{j,\ell})^r)$.

Note that (a_1, a_2) is sufficient for decryption, while $\vec{z}_{k+1}, \cdots, \vec{z}_h$ can help the derivation of the trapdoor for $(\mathsf{ID}_1, \cdots, \mathsf{ID}_k, \mathsf{ID}_{k+1})$ for any n-bit string ID_{k+1} and $k+1 \leq h$. To generate $d_{\overrightarrow{ID}||\mathsf{ID}_r}$ parse $d_{\overrightarrow{ID}} = (a_1, a_2, (z_{k+1}, z_{k+1,1}, \cdots, z_{k+1,\ell}), \cdots, (z_h, z_{h,1}, \cdots, z_{h,\ell}))$ and parse ID_r as ℓ blocks $(\mathsf{ID}_{r,1}, \cdots, \mathsf{ID}_{r,\ell})$ where each block is of length n/ℓ bits, pick $t \in_R \mathbb{Z}_p^*$ and output

$$d_{\overrightarrow{ID}||\mathsf{ID}_r} = (a_1 \cdot z_{k+1} \prod_{i=1}^{\ell} (z_{k+1,i})^{\mathsf{ID}_{r,i}} \cdot (\prod_{j=1}^{k+1} F_{\vec{U}_j}(\mathsf{ID}_j))^t, a_2 \cdot g^t, \vec{z}_{k+2} \cdot (\vec{U}_{k+2})^t \cdots, \vec{z}_h \cdot (\vec{U}_h)^t$$

where the multiplication of two vectors are defined component-wise, i.e. $\vec{z}_j \cdot \vec{\nu}_j = (z_j \cdot \nu_j, z_{j,1} \cdot \nu_{j,1}, \cdots, z_{j,\ell} \cdot \nu_{j,\ell})$. $d_{\overrightarrow{ID}}$ becomes shorter as the length of $\overrightarrow{ID}$ increases.

$\mathsf{KGen}()$: Pick $\mathsf{sk} \in_R \mathbb{Z}_p^*$, return $\mathsf{pk} = (X, Y) = (g^{\mathsf{sk}}, g_1^{\mathsf{sk}})$ and sk as the key pair.

$\mathsf{Enc}(m, \overrightarrow{ID}, \mathsf{pk})$: To encrypt $m \in \mathbb{G}_T$ for $\overrightarrow{ID} = (\mathsf{ID}_1, \cdots, \mathsf{ID}_k)$ where $k \leq h$, parse pk as (X, Y), then check that it is a valid public key by verifying[4] that $\hat{e}(X, g_1) = \hat{e}(g, Y)$. If equality holds, pick $s \in_R \mathbb{Z}_p^*$ and compute

$$C = (C_1, C_2, \tau, \sigma) = (m \cdot \hat{e}(Y, g_2)^s, \prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j)^s, g^s, F_{\vec{V}}(w)^s)$$

where $w = H(C_1, C_2, \tau, \overrightarrow{ID}, \mathsf{pk})$.

$\mathsf{Dec}^S(C, d_{\overrightarrow{ID}})$: Parse C as (C_1, C_2, τ, σ), and $d_{\overrightarrow{ID}}$ as $(a_1, a_2, \cdots)$. First check if $\hat{e}(\tau, \prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j) \cdot F_{\vec{V}}(w')) = \hat{e}(g, C_2 \cdot \sigma)$ where $w' = H(C_1, C_2, \tau, \overrightarrow{ID}, \mathsf{pk})$. Return $\perp$ if inequality holds or any parsing is not possible, otherwise pick $t \in_R \mathbb{Z}_p^*$ and return

$$D = (D_1, D_2, D_3) = (a_1 \cdot F_{\vec{V}}(w')^t, a_2, g^t).$$

$\mathsf{Dec}^U(C, \mathsf{sk}, D)$: Parse C as (C_1, C_2, τ, σ) and check if $\hat{e}(\tau, \prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j) \cdot F_{\vec{V}}(w')) = \hat{e}(g, C_2 \cdot \sigma)$ where $w' = H(C_1, C_2, \tau, \overrightarrow{ID}, \mathsf{pk})$. If equality does not hold or parsing is not possible, return $\perp_C$. Next, parse D as (D_1, D_2, D_3) and check if $\hat{e}(g, D_1) = \hat{e}(g_1, g_2)\hat{e}(D_2, \prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j))\hat{e}(D_3, F_{\vec{V}}(w'))$[5]. If equality does not hold or parsing is not possible, return $\perp_D$. Otherwise, return

$$m \leftarrow C_1 \cdot \left(\frac{\hat{e}(C_2, D_2)\hat{e}(\sigma, D_3)}{\hat{e}(\tau, D_1)} \right)^{\mathsf{sk}}.$$

[4] One pairing computation can be saved by a trick adopted in [18]: pick $\xi \in_R \mathbb{Z}_p^*$ and compute $C_1 = m \cdot \hat{e}(Y, g_2 \cdot g^{\xi})^s / \hat{e}(X, g_1^{s\xi})$.

[5] The same trick for minimizing the number of pairing computations involved in checking the ciphertext and the token can be incorporated to the final decryption step. The modified decryption algorithm only uses 4 pairing computations; however, it gives a random message (instead of an invalid flag $\perp$) for an invalid ciphertext.

4.3 Analysis

Similar to [4], the ciphertext size of our scheme is independent of the hierarchy length. This is also beneficial when it is used as a TRE (see Section 5.5).

In the concrete SMCLE scheme of Chow, Boyd and González Nieto [14], partial decryption uses the pairing function $\hat{e}(\cdot,\cdot)$ to pair part of the ciphertext and the ID-based private key. To make this partial decryption result verifiable requires turning a generic interactive proof-of-knowledge non-interactive. Our scheme employs a different technique such that the token generated by the partial decryption is publicly and non-interactively verifiable.

Our scheme's security is asserted by Theorem 2; Appendix A contains a proof.

Theorem 2 *Our scheme is secure against Type-I attack and Type-II attack (Definition 3 and 4) if h-wDHI problem and 3-DDH problem is intractable.*

5 Applying General Certificateless Encryption to TRE

5.1 Syntax of Timed-Release Encryption

For ease of discussion, consider only 1-level of time-identifiers as in [7]. It can be shown that our results hold for an h-level analog.

Definition 8. *A TRE scheme for time-identifiers of length n (n is a polynomially-bounded function) is defined by the following sextuple of* PPT *algorithms:*

- Setup *(run by the server) is a probabilistic algorithm which takes a security parameter* 1^λ*, outputs a master secret key* Msk*, and the global parameters* Pub*. We assume that* λ *and* $n = n(\lambda)$ *are implicit in* Pub *and all other algorithms take* Pub *implicitly as an input.*
- Extract *(run by the server) is a possibly probabilistic algorithm which takes the master secret key* Msk *and a string* $\mathsf{ID} \in \{0,1\}^n$*, outputs a trapdoor key* d_{ID} *associated with the identifier* ID*.*
- KGen *(run by a user) is a probabilistic algorithm which generates a public/private key pair* $(\mathsf{pk}_u, \mathsf{sk}_u)$*.*
- Enc *(run by a sender) is a probabilistic algorithm which takes a message* m *from some implicit message space, an identifier* $\mathsf{ID} \in \{0,1\}^n$*, and the receiver's public key* pk_u *as input, returns a ciphertext* C*.*
- Dec^S *(run by any one who holds the trapdoor, either a SEM or a receiver) is a possibly probabilistic algorithm which takes a ciphertext* C *and a trapdoor key* d_{ID} *as input, returns either a token* D *which can be seen as a partial decryption of* C*, or an invalid flag* $\perp$ *(which is not in the message space).*
- Dec^U *(run by a receiver) is a possibly probabilistic algorithm which takes the ciphertext* C*, the receiver's secret key* sk_u *and a token* D *as input, returns either the plaintext, an invalid flag* $\perp_D$ *denoting* D *is an invalid token, or an invalid flag* $\perp_C$ *denoting the ciphertext is invalid.*

For correctness, we require that $\mathsf{Dec}^U(C, \mathsf{sk}, \mathsf{Dec}^S(C, \mathsf{Extract}(\mathsf{Msk}, \mathsf{ID}))) = m$ *for all* $\lambda \in \mathbb{N}$*, all* $(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$*, all* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}$*, all message* m*, all identifier* ID *in* $\{0,1\}^n$ *and all* $C \xleftarrow{\$} \mathsf{Enc}(m, \mathsf{ID}, \mathsf{pk})$.

5.2 Timed-Release Encryption from Certificateless Encryption

Given a SMCLE scheme $\{\mathcal{SMC}.\mathsf{Setup}, \mathcal{SMC}.\mathsf{Extract}, \mathcal{SMC}.\mathsf{KGen}, \mathcal{SMC}.\mathsf{Enc}, \mathcal{SMC}.\mathsf{Dec}^S, \mathcal{SMC}.\mathsf{Dec}^U\}$, a TRE scheme $\{\mathcal{TRE}.\mathsf{Setup}, \mathcal{TRE}.\mathsf{Extract}, \mathcal{TRE}.\mathsf{KGen}, \mathcal{TRE}.\mathsf{Enc}, \mathcal{TRE}.\mathsf{Dec}^S, \mathcal{TRE}.\mathsf{Dec}^U\}$ can be built as below.

$\mathcal{TRE}.\mathsf{Setup}(1^\lambda, n)$: Given a security parameter λ and the length of the time-identifier n, execute $(\mathsf{Msk}, \mathsf{Pub}) \leftarrow \mathcal{SMC}.\mathsf{Setup}(1^\lambda, n)$, retain Msk as the master secret key and publish Pub as the global parameters.

$\mathcal{TRE}.\mathsf{Extract}(\mathsf{Msk}, \mathsf{ID})$: For a time-identifier $\mathsf{ID} \in \{0,1\}^n$, the time-server returns $d_{\mathsf{ID}} \leftarrow \mathcal{SMC}.\mathsf{Extract}(\mathsf{Msk}, \mathsf{ID})$.

$\mathcal{TRE}.\mathsf{KGen}()$: Return $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{SMC}.\mathsf{KGen}()$ as the user's key pair.

$\mathcal{TRE}.\mathsf{Enc}(m, \mathsf{ID}, \mathsf{pk})$: To encrypt $m \in \mathbb{G}_T$ for pk under the time $\mathsf{ID} \in \{0,1\}^n$, return $\mathcal{SMC}.\mathsf{Enc}(m, \mathsf{ID}, \mathsf{pk})$, which may be $\perp$ if pk is an invalid public key.

$\mathcal{TRE}.\mathsf{Dec}^S(C, d_{\mathsf{ID}})$: To partially decrypt C by a time-dependent trapdoor d_{ID}, return $D \leftarrow \mathcal{SMC}.\mathsf{Dec}^S(C, d_{\mathsf{ID}})$.

$\mathcal{TRE}.\mathsf{Dec}^U(C, \mathsf{sk}, D)$: To decrypt C by the secret key sk and the token D, just return $\mathcal{SMC}.\mathsf{Dec}^U(C, \mathsf{sk}, D)$.

Theorem 3 *If* $\mathcal{SMC}$ *is an* 1*-level SMCLE scheme which is CCA-secure against Type-I adversary (Definition 3),* $\mathcal{TRE}$ *is CCA-secure against Type-I adversary.*

Theorem 4 *If* $\mathcal{SMC}$ *is an* 1*-level SMCLE scheme which is CCA-secure against Type-II adversary (Definition 4),* $\mathcal{TRE}$ *is CCA-secure against Type-II adversary.*

Proof. The security models of TRE can be found in Appendix B. We prove by contradiction. Suppose $\mathcal{A}$ is a Type-X adversary such that $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA}'-\mathsf{X}}(\lambda) = 1] - \frac{1}{2}| > \epsilon$, we construct an adversary $\mathcal{B}$ with $|\Pr[\mathbf{Exp}_{\mathcal{B}}^{\mathsf{CCA}-\mathsf{X}}(\lambda) = 1] - \frac{1}{2}| > \epsilon$ in the face of a SMCLE challenger $\mathcal{C}$ where the running times of $\mathcal{B}$ and $\mathcal{A}$ are equal.

Setup: When $\mathcal{C}$ gives $\mathcal{B}$ $(\mathsf{Pub}, \mathsf{Aux})$, $\mathcal{B}$ just forwards it to $\mathcal{A}$. The public key to be passed to $\mathcal{A}$ is either chosen from the a set of public key in Aux (in Type-II game), or chosen by $\mathcal{B}$ itself (in Type-I game).

First Phase of Queries: $\mathcal{B}$ forwards every request of $\mathcal{A}$ to the oracles of its own challenger $\mathcal{C}$. From the description of $\mathcal{TRE}$, we can see that every legitimate oracle query made by $\mathcal{A}$ can be answered faithfully.

Challenge: When $\mathcal{A}$ gives $\mathcal{B}$ $(m_0, m_1, \mathsf{pk}^*, \mathsf{ID}^*)$, $\mathcal{B}$ just forwards it to $\mathcal{C}$.

Second Phase of Queries: Again, $\mathcal{B}$ just forwards every request of $\mathcal{A}$ to the oracles of its own challenger $\mathcal{C}$. From the description of $\mathcal{TRE}$, it is easy to see that every oracle query which does not violate the restriction enforced by $\mathcal{A}$ also does not violate the restriction enforced by $\mathcal{C}$.

Output: Finally, $\mathcal{A}$ outputs a bit b, $\mathcal{B}$ forwards it to $\mathcal{C}$ as its own answer. The probability for $\mathcal{A}$ to win the TRE experiment simulated by $\mathcal{B}$ is equal to the probability for $\mathcal{B}$ to win the SMCLE game played against $\mathcal{C}$. It is easy to see that the running times of $\mathcal{A}$ and $\mathcal{B}$ are the same. □

These theorems show that the scheme presented in section 4 can be instantiated as a TRE scheme without a random oracle.

5.3 Certificateless Encryption from Timed-Release Encryption

One may expect that a general CLE can be constructed from any TRE. The usage of time-identifiers, however, is only one specific instantiation of the timed-release idea. Other formulations of TRE, different from Definition 8, exist; for example, in [9], time is captured by the number of repeated computations of one-way hash function. Also, the notion of CLE supports an exponential number of arbitrary identifiers[6], so a CLE scheme cannot be realized by a TRE if the total number of time periods supported is too few.

There is an important difference in the definitions of security between CLE and TRE: the public keys in TRE are *certified* while there is no certification in CLE, so public keys can be chosen adversarially. Typically in TRE [3, 8, 12, 19, 24], a single public key is *given* to the adversary as the target of attack. However, the non-standard TRE formulation in [7] does allow uncertified public keys.

5.4 Security-Mediator in Timed-Release Encryption

The introduction of a security-mediator to the TRE paradigm gives a new business model for the time-server due to the support for partial decryption. Traditional TRE allows the time-server to release only a system-wide time-dependent trapdoor. The time-server can charge for each partial decryption request of a ciphertext by the time-dependent trapdoor; the partial decryption of one ciphertext would not help the decryption of any other ciphertext.

5.5 Time Hierarchy

Since each identifier corresponds to a single time period, the server must publish t private keys once t time-periods have passed. The amount of data that must be posted can be reduced given a hierarchical CLE by using the Canetti, Halevi and Katz (CHK) forward secure encryption [6] in reverse [4]. For a total of T

[6] Even though the scheme may be insecure when more than a polynomial number of trapdoors are compromised by a single adversary.

time periods, the CHK framework is set up as a tree of depth $\log T$. To encrypt a message for time $t < T$, the time identifier is the CHK identifier for time period $T-t$. Release of trapdoor is done in the same manner: the private key for the time period $T - t$ is released on the t^{th} time period. This single private key enables anyone to derive the private keys for CHK time periods $T - t, T - t + 1, \cdots, T$, so the user can obtain trapdoors for times $1, \cdots, t$. This trick enables the server to publish only a single private key of $O(\log^2 T)$ group elements at any time.

6 Conclusions

Cryptographers seek and try to achieve the strongest possible security definition. Previous models of certificateless encryption (CLE) were too restrictive: they could not give the desired security properties when instantiated as timed-release encryption (TRE). Our generalized CLE model supports the requirements of TRE; all future CLE proposals in our general model automatically give secure TRE schemes. Our model is defined against full-identifier extraction, decryption under arbitrary public key, and partial decryption, to achieve strong security. Our concrete scheme yields the first strongly-secure (hierarchical) security-mediated CLE and the first TRE in the standard model.

References

1. Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless Public Key Cryptography. In *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 452–473. Springer, 2003. Full version at `http://eprint.iacr.org/2003/126`.
2. Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Certificateless Public Key Encryption Without Pairing. In *Information Security Conference, ISC 2005*, volume 3650 of *LNCS*, pages 134–148. Springer, 2005.
3. Ian F. Blake and Aldar C-F. Chan. Scalable, Server-Passive, User-Anonymous Timed Release Cryptography. In *ICDCS 2005*, pages 504–513. IEEE Computer Society, 2005.
4. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.
5. Xavier Boyen, Qixiang Mei, and Brent Waters. Direct Chosen Ciphertext Security from Identity-based Techniques. In *ACM CCS 2005*, pages 320–329, 2005.
6. Ran Canetti, Shai Halevi, and Jonathan Katz. A Forward-Secure Public-Key Encryption Scheme. *Journal of Cryptology*, 20(3):265–294, 2007.
7. Julien Cathalo, Benoît Libert, and Jean-Jacques Quisquater. Efficient and Non-interactive Timed-Release Encryption. In *Information and Communications Security, ICICS 2005*, volume 3783 of *LNCS*, pages 291–303. Springer, 2005.
8. Konstantinos Chalkias, Dimitrios Hristu-Varsakelis, and George Stephanides. Improved Anonymous Timed-Release Encryption. In *ESORICS 2007*, volume 4734 of *LNCS*, pages 311–326. Springer, 2007.
9. Konstantinos Chalkias and George Stephanides. Timed Release Cryptography from Bilinear Pairings Using Hash Chains. In *Communications and Multimedia Security, CMS 2006*, volume 4237 of *LNCS*, pages 130–140. Springer, 2006.

10. Sanjit Chatterjee and Palash Sarkar. New Constructions of Constant Size Ciphertext HIBE Without Random Oracle. In *Information Security and Cryptology, ICISC 2006*, volume 4296 of *LNCS*, pages 310–327. Springer, 2006.
11. Sanjit Chatterjee and Palash Sarkar. On (Hierarchical) Identity Based Encryption Protocols with Short Public Parameters (With an Exposition of Waters' Artificial Abort Technique). Cryptology ePrint Archive, Report 2006/279, 2006.
12. Jung Hee Cheon, Nicholas Hopper, Yongdae Kim, and Ivan Osipkov. Timed-Release and Key-Insulated Public Key Encryption. In *Financial Cryptography and Data Security, FC 2006*, volume 4107 of *LNCS*, pages 191–205. Springer, 2006.
13. Sherman S. M. Chow. Token-Controlled Public Key Encryption in the Standard Model. In *Information Security Conference, ISC 2007*, volume 4779 of *LNCS*, pages 315–332. Springer, 2007.
14. Sherman S. M. Chow, Colin Boyd, and Juan Manuel González Nieto. Security-Mediated Certificateless Cryptography. In *Public Key Cryptography - PKC 2006*, volume 3958 of *LNCS*, pages 508–524. Springer, 2006.
15. Sherman S.M. Chow. Certificateless Encryption. In *Identity-Based Cryptography*. IOS Press, 2008. To appear.
16. Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional Oblivious Transfer and Timed-Release Encryption. In *EUROCRYPT '99*, volume 1592 of *LNCS*, pages 74–89. Springer, 1999.
17. Alexander W. Dent. A Survey of Certificateless Encryption Schemes and Security Models. Cryptology ePrint Archive, Report 2006/211, 2006.
18. Alexander W. Dent, Benoit Libert, and Kenneth G. Paterson. Certificateless Encryption Schemes Strongly Secure in the Standard Model. In *Public Key Cryptography - PKC 2008*, volume 4939 of *LNCS*, pages 344–359. Springer, 2008. Full version at `http://eprint.iacr.org/2007/121`.
19. Alexander W. Dent and Qiang Tang. Revisiting the Security Model for Timed-Release Public-Key Encryption with Pre-Open Capability. In *Information Security Conference, ISC 2007*, volume 4779 of *LNCS*, pages 158–174. Springer, 2007.
20. Craig Gentry. Practical Identity-Based Encryption Without Random Oracles. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464. Springer, 2006.
21. Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.
22. Dimitrios Hristu-Varsakelis, Konstantinos Chalkias, and George Stephanides. Low-cost Anonymous Timed-Release Encryption. In *Symposium on Information Assurance and Security*, pages 77–82. IEEE Computer Society, 2007.
23. Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, and Xiaotie Deng. Certificateless Signature: A New Security Model and An Improved Generic Construction. *Designs, Codes and Cryptography*, 42(2):109–126, 2007.
24. Yong Ho Hwang, Dae Hyun Yum, and Pil Joong Lee. Timed-Release Encryption with Pre-open Capability and Its Application to Certified E-mail System. In *Information Security Conference, ISC 2005*, volume 3650 of *LNCS*, pages 344–358. Springer, 2005.
25. Junzuo Lai and Weidong Kou. Self-Generated-Certificate Public Key Encryption Without Pairing. In *Public Key Cryptography, PKC 2007*, volume 4450 of *LNCS*, pages 476–489. Springer, 2007.
26. Joseph K. Liu, Man Ho Au, and Willy Susilo. Self-Generated-Certificate Public Key Cryptography and Certificateless Signature / Encryption Scheme in the Standard Model. In *ASIACCS 2007*. ACM, 2007.
27. Deholo Nali, Carlisle M. Adams, and Ali Miri. Hierarchical Time-based Information Release. *International Journal of Information Security*, 5(2):92–104, 2006.

28. Jong Hwan Park, Kyu Young Choi, Jung Yeon Hwang, and Dong Hoon Lee. Certificateless Public Key Encryption in the Selective-ID Security Model (Without Random Oracles). In *Pairing-Based Cryptography 2007*, volume 4575 of *LNCS*, pages 60–82. Springer, 2007.
29. Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock Puzzles and Timed-release Crypto. Technical Report MIT/LCS/TR-684, Massachusetts Institute of Technology, 1996.
30. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.
31. Yinxia Sun, Futai Zhang, and Joonsang Baek. Strongly Secure Certificateless Public Key Encryption Without Pairing. In *Cryptology and Network Security, CANS, 2007*, volume 4856 of *LNCS*, pages 194–208. Springer, 2007.
32. Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

A Formal Security Proof for Our Proposed Construction

We now define a series of games where each one is an interactive game between a simulator $\mathcal{S}$ and an adversary $\mathcal{A}$, which is either an insider attacker (Type-I adversary) or a curious server (Type-II adversary), depending on the allowed queries. The skeleton of the proof is based on the proof given in [18].

Game 1 (The Original Game). This game is the one played between a simulator $\mathcal{S}$ and an adversary $\mathcal{A}$ as specified in the experiment $\mathbf{Exp}^{\mathsf{CCA-X}}$. We use the following notation: For the queries, let $\mathcal{T} = \{\overrightarrow{ID}_1, \cdots, \overrightarrow{ID}_{q_E}\}$ denote the trapdoors extraction queries and $\mathcal{W} = \{w_1, \cdots, w_{q_D}\}$ be the set of strings involved in decryption queries where $w_j = H(C_1, C_2, \tau, \overrightarrow{ID}_j, \mathsf{pk}_j)$. For the challenges, let $\overrightarrow{ID}^*$ and pk^* denote the challenge identifier and the challenge public key respectively, and let $C^* = (C_1^*, C_2^*, \tau^*, \sigma^*)$ be the returned challenge ciphertext and let $w^* = H(C_1^*, C_2^*, \tau^*, \overrightarrow{ID}^*, \mathsf{pk}^*)$. The random bit ι is chosen by $\mathcal{S}$ in order to select which message is encrypted.

Game 2 (Change of Public Parameters). Let $Z_i = (g)^{\alpha^i}$, $1 \leq i \leq h+1$. This game is the same as Game 1 except that the generation of the parameters is changed. $\mathcal{S}$ picks $\alpha, \beta \in_R \mathbb{Z}_p$, and set $g_1 = Z_1$, $g_2 = Z_h \cdot g^\beta$.

If $mode = \text{II}$, $\mathcal{A}$ should obtain $\mathsf{Aux} = \mathsf{Msk}, \{\mathsf{pk}_1^*, \cdots \mathsf{pk}_{q_K}^*\}$ from $\mathcal{S}$. $\mathcal{S}$ computes $\mathsf{Msk} = \alpha^h + \beta$. For public keys, $\mathcal{S}$ randomly chooses θ_i from $\mathbb{Z}_p$, and computes $\mathsf{pk}_i^* = (X_i^*, Y_i^*) = (g^{\theta_i}, (g^\alpha)^{\theta_i}), \forall i \in \{1, \cdots, q_K\}$.

The simulator also changes the vectors as follows. Let ρ_u and ρ_v be integers such that $\rho_u(n+1) < p$ and $\rho_v(n+1) < p$. The exact choices of ρ_u and ρ_u will be determined later. The simulator randomly selects

- $\kappa_{u_1}, \cdots, \kappa_{u_n}, \kappa_v$ from $\{0, \cdots, \ell(n^{1/\ell} - 1)\}$,
- h many $(\ell+1)$-length vectors $\vec{x}_1, \cdots, \vec{x}_h$ from $\mathbb{Z}_{\rho_u}$, where each $\vec{x}_j = (x'_j, x_{j,1}, \cdots, x_{j,\ell})$.

- h many $(\ell+1)$-length vectors $\vec{y}_1, \cdots, \vec{y}_h$ from $\mathbb{Z}_p$, where each $\vec{y}_j = (y'_j, y_{j,1}, \cdots, y_{j,\ell})$.
- $(x'_v, x_{v,1}, \cdots, x_{v,n})$ from $\mathbb{Z}_{\rho_v}^{n+1}$
- $(y'_v, y_{v,1}, \cdots, y_{v,n})$ from $\mathbb{Z}_p^{n+1}$.

The hash keys for the identifier-based derivation, for $1 \le j \le h$, are set as:

$$u'_j = Z_{h-j+1}^{(p+\rho_u\kappa_j - x'_j)} \cdot g^{y'_j}, \quad u_{j,i} = Z_{h-j+1}^{-x'_{j,i}} \cdot g^{y_{j,i}} \text{ for } 1 \le i \le \ell.$$

The hash key for the ciphertext validity is set as (note that $g_2 = Z_h \cdot g^{\beta}$):

$$v' = g_2^{(p+\rho_v\kappa_v - x'_v)} \cdot g^{y'_v}, \qquad v_i = g_2^{-x_{v,i}} g^{y'_{v,i}} \text{ for } 1 \le i \le n.$$

Define the below functions which take $\mathsf{ID}_j = (\mathsf{ID}_{j,1}, \cdots, \mathsf{ID}_{j,\ell})$ or $w = b_1 \cdots b_n$.

$$J_{u_1}(\mathsf{ID}_1) = p + \rho_u\kappa_1 - x'_1 - \sum_{i=1}^{\ell} x_{1,i}\mathsf{ID}_{1,i},\ K_{u_1}(\mathsf{ID}_1) = y'_1 + \textstyle\sum_{1=1}^{\ell} y_{1,i}\mathsf{ID}_{h,i},$$

$$\vdots \qquad\qquad \vdots$$

$$J_{u_h}(\mathsf{ID}_h) = p + \rho_u\kappa_h - x'_h - \sum_{i=1}^{\ell} x_{h,i}\mathsf{ID}_{h,i},\ K_{u_h}(\mathsf{ID}_h) = y'_h + \textstyle\sum_{1=1}^{\ell} y_{h,i}\mathsf{ID}_{h,i},$$

$$J_v(w) = p + \rho_v\kappa_v - x'_v - \sum_{i=1}^{\ell} x_{v,i}b_i, \qquad K_v(w) = y'_v + \textstyle\sum_{1=1}^{\ell} y_{v,i}b_i,$$

The settings above give

$$F_{\vec{U}_j}(\mathsf{ID}_j) = u'_j \prod_{i=1}^{\ell} u_{j,i}^{\mathsf{ID}_{j,i}} = Z_{h-j+1}^{J_{u_j}(\mathsf{ID}_j)} \cdot g^{K_{u_j}(\mathsf{ID}_j)}, j \in \{1, \cdots, h\}$$

$$F_{\vec{V}}(w) = v' \prod_{j=1}^{n} {v_j}^{b_j} = g_2^{J_v(w)} \cdot g^{K_v(w)}$$

These changes do not change the distribution of the public parameters, so we have $\Pr[S_2] = \Pr[S_1]$.

Game 3 (Elimination of Hash Collisions). The simulator aborts and assumes $\mathcal{A}$ outputs a random bit in this game if $\mathcal{A}$ submits a decryption query $(C, \overrightarrow{ID}, \mathsf{pk} = (g^{\mathsf{sk}}, g_1^{\mathsf{sk}}))$ for a well-formed ciphertext $C = (C_1, C_2, \tau, \sigma)$ where $w = H(C_1, C_2, \tau, \overrightarrow{ID}, \mathsf{pk})$ is either equal to the same value as a previously submitted ciphertext or w^* of the challenge ciphertext.

For such a decryption query to be legal, we have $C \neq C^*$ or $(\overrightarrow{ID}, \mathsf{pk}) \neq (\overrightarrow{ID}^*, \mathsf{pk}^*)$. In either case, this implies a collision for H, which means we can construct an adversary $\mathcal{C}$ against the collision resistance of H such that $|\Pr[S_3] - \Pr[S_2]| \le Adv_{\mathcal{C}}^{\mathrm{CR}}(\lambda)$.

Game 4 (Preparation for the Simulation of the Challenge Ciphertext). Let $\overrightarrow{ID}^* = (\mathsf{ID}_1^*, \cdots, \mathsf{ID}_k^*)$ where $k \leq h$. This time $\mathcal{S}$ aborts if $J_{u_j}(\mathsf{ID}_j^*) \neq 0 \bmod p$ for any $j \in \{1, \cdots, k\}$ or $J_v(w^*) \neq 0 \bmod p$.

Since the values determining these functions are information theoretically hidden from $\mathcal{A}$, such ID^* and w^* can only be produced by chance. Therefore

$$\begin{aligned} & \Pr[J_v(w^*) = 0 \bmod p] \\ = \; & \Pr[J_v(w^*) = 0 \bmod p | J_v(w^*) = 0 \bmod \rho_v] \cdot \Pr[J_v(w^*) = 0 \bmod \rho_v] \\ = \; & \frac{1}{\rho_v(n+1)} \end{aligned}$$

Unless $\mathcal{S}$ aborts, Game 3 and Game 4 are identical and we have $|\Pr[S_4] - \Pr[S_3]| \leq \frac{1}{(\rho_u)^h \rho_v (\ell+1)^{h+1}}$ by a similar computation ($n \geq \ell$). The significance of this extra abort condition will be manifested in Game 8.

Game 5 (Artificial Abort for Consistent View of Adversary). Now $\mathcal{S}$ aborts if $J_{u_1}(\mathsf{ID}'_1) = \cdots = J_{u_k}(\mathsf{ID}'_k) = 0 \bmod \rho_u$ for some $\overrightarrow{ID}' = (\mathsf{ID}'_1, \cdots, \mathsf{ID}'_k) \in \mathcal{T}$ or $J_v(w') = 0 \bmod \rho_v$ for some $w' \in \mathcal{W}$.

Since $\mathcal{A}$'s power is dependent on the extraction and decryption queries, the above abort event is not independent of S_4, and we cannot relate the probability of S_4 and S_5 in a similar way as before.

This problem can be circumvented by the "re-normalization" technique due to Waters [32], such that "artificial aborts" are added to make sure that the probability of aborts is exactly equal to some negligible upper bound for the probability that E occurs for any set of oracle queries.

Conditioning on the event $J_v(w^*) = 0 \bmod p$, the theoretical lower bound of $\Pr[J_v(w^*) \neq 0 \bmod p]$ is $(1 - \frac{q_D}{\rho_v})$. Setting $\rho_v = 2q_D$ and will make it bounded by $1/2$. On the other hand, a lower bound on the probability for the first event is $\frac{1}{2(4\ell q_E 2^{n/\ell})^h}$ by setting $\rho_u = 4q_E$ [10].

We estimate the probability that $\mathcal{A}$'s oracle queries will cause $\mathcal{S}$ to abort by repeatedly sampling values determining $J_{u_1}(\cdot), \cdots, J_{u_h}(\cdot), J_v(\cdot)$. This would not involve re-running $\mathcal{A}$ as $\mathcal{A}$'s view (of the public parameters) remains unchanged by assuming y's are changing accordingly. Waters [32] has shown that a polynomial number of trials is sufficient to give an estimate of the abort probability η to within a negligible error term.

If $\mathcal{S}$ did not abort, we force an artificial abort with probability

$$(\eta - 1/(4(4\ell q_E 2^{n/\ell})^h))/\eta,$$

and $\mathcal{S}$ will abort with probability sufficiently close to $\frac{1}{4(4\ell q_E 2^{n/\ell})^h}$. Now we can say $\Pr[S_5] = \Pr[S_4]/4(4\ell q_E 2^{n/\ell})^h$. An exposition of Waters' technique can be found at [11].

Game 6 (Simulation of Extraction and Decryption). This game changes the simulation of all $\mathcal{A}$'s queries for trapdoor extractions, partial decryptions,

and complete decryptions. We will have $\Pr[S_6] = \Pr[S_5]$.

Trapdoor extraction: For trapdoor key extraction query of $\overrightarrow{ID} = (\mathsf{ID}_1, \cdots, \mathsf{ID}_k)$ where $k \leq h$. Let $j' \in \{1, \cdots, k\}$ be a minimum one such that $J_{u_{j'}}(\mathsf{ID}_{j'}) \neq 0$. There exists such a j' or $\mathcal{S}$ has aborted in Game 5. $\mathcal{S}$ needs to return $d_{\overrightarrow{ID}} = (a_1, a_2, \vec{z}_{k+1}, \cdots, \vec{z}_h)$.

We first show how to compute $a_{1|j'}$, a "trapdoor for only $\mathsf{ID}_{j'}$" (without any appearance of any elements from other levels); then we will show how to compute a trapdoor $(a_1, a_2, \vec{z}_{k+1}, \cdots, \vec{z}_h)$ that matches the same implicit random factor used in $a_{1|j'}$. Recall that $F_{\vec{U}_{j'}}(\mathsf{ID}_{j'}) = Z_{h-j'+1}^{J_{u_{j'}}(\mathsf{ID}_{j'})} \cdot g^{K_{u_{j'}}(\mathsf{ID}_{j'})}$. $\mathcal{S}$ picks $r \in \mathbb{Z}_p^*$ and computes

$$a_{1|j'} = (Z_1^{\beta} \cdot Z_{j'}^{-\frac{K_{u_{j'}}(\mathsf{ID}_{j'})}{J_{u_{j'}}(\mathsf{ID}_{j'})}}) \cdot (Z_{h-j'+1}^{J_{u_{j'}}(\mathsf{ID}_{j'})} \cdot g^{K_{u_{j'}}(\mathsf{ID}_{j'})})^r$$

The second component of $a_{1|j'}$ is only for randomization. We will show the first component of $a_{1|j'}$ is in the form of $g_2^{\alpha}(F_{\vec{U}_{j'}}(\mathsf{ID}_{j'}))^{-\frac{\alpha^{j'}}{J_{u_{j'}}(\mathsf{ID}_{j'})}}$, which means $a_{1|j'}$ is in the form of $g_2^{\alpha}(F_{\vec{U}_{j'}}(\mathsf{ID}_{j'}))^{\tilde{r}}$ where $\tilde{r} = r - \frac{\alpha^{j'}}{J_{u_{j'}}(\mathsf{ID}_{j'})}$.

$$\begin{aligned}
& g_2^{\alpha}(F_{\vec{U}_{j'}}(\mathsf{ID}_{j'}))^{-\frac{\alpha^{j'}}{J_{u_{j'}}(\mathsf{ID}_{j'})}} \\
&= (Z_h \cdot g^{\beta})^{\alpha}(Z_{h-j'+1}^{J_{u_{j'}}(\mathsf{ID}_{j'})} \cdot g^{K_{u_{j'}}(\mathsf{ID}_{j'})})^{-\frac{\alpha^{j'}}{J_{u_{j'}}(\mathsf{ID}_{j'})}} \\
&= Z_{h+1} \cdot Z_1^{\beta} \cdot Z_{h+1}^{-\frac{J_{u_{j'}}(\mathsf{ID}_{j'})}{J_{u_{j'}}(\mathsf{ID}_{j'})}} \cdot Z_{j'}^{-\frac{K_{u_{j'}}(\mathsf{ID}_{j'})}{J_{u_{j'}}(\mathsf{ID}_{j'})}} \\
&= Z_1^{\beta} \cdot Z_{j'}^{-\frac{K_{u_{j'}}(\mathsf{ID}_{j'})}{J_{u_{j'}}(\mathsf{ID}_{j'})}}
\end{aligned}$$

To compute $a_1 = g_2^{\alpha} \cdot (\prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j))^{\tilde{r}}$, $\mathcal{S}$ needs to compute $F_{\vec{U}_j}(\mathsf{ID}_j)^{\tilde{r}} = (Z_{h-j+1}^{J_{u_j}(\mathsf{ID}_j)})^{\tilde{r}} \cdot (g^{K_{u_j}(\mathsf{ID}_j)})^{\tilde{r}}$ for $j \neq j'$. We would like to compute it without knowing α and Z_{h+1}, but with the help of $(Z_1, \cdots, Z_h)$. Now the only difficulty comes from the fact that $\alpha^{j'}$ in $\tilde{r}$ is unknown. Note that the second term $(g^{K_{u_j}(\mathsf{ID}_j)})^{\alpha^{j'}}$ can be computed from $Z_{j'}$. We can see how the first term can be obtained by considering two different cases.

1. $j < j'$: $J_{u_j}(\mathsf{ID}_j) = 0$ by the choice of j'.
2. $j > j'$: $Z_{h-j+1}^{\alpha^{j'}} = Z_{h+1-(j-j')}$, note that $1 \leq j - j' \leq h - 1$.

By similar reasoning, since $k + 1 > j'$, it is easy to see that $\vec{z}_{k+1}, \cdots, \vec{z}_h$ can also be computed from $(Z_1, \cdots, Z_h)$. This completes the simulation of the trapdoor queries.

SEM partial decryption: $\mathcal{S}$ performs the usual validity checking to reject any invalid ciphertext C that is purported to be encrypted under $\overrightarrow{ID}$ and pk. For

decrypting a valid ciphertext with hash w by the trapdoor of $\overrightarrow{ID}$, if $d_{\overrightarrow{ID}} = (a_1, a_2, \cdots)$ is computable by $\mathcal{S}$, it is easy to generate $(a_1 F_{\overrightarrow{V}}(w)^t, a_2, g^t)$ for a random $t \in \mathbb{Z}_p^*$.

$\mathcal{S}$ cannot generate the trapdoor for $d_{\overrightarrow{ID}}$ only if $J_{u_1}(\mathsf{ID}_1) = \cdots = J_{u_k}(\mathsf{ID}_k) = 0 \bmod \rho_u$. Note that $J_v(w) \neq 0 \bmod \rho_v$ or $\mathcal{S}$ has aborted in Game 5. Under this condition, $\mathcal{S}$ can generate the token similar to the generation of the trapdoor before. Recall that $F_{\overrightarrow{V}}(w) = (Z_h \cdot g^\beta)^{J_v(w)} \cdot g^{K_v(w)}$, we have

$$\begin{aligned}
& g_2^\alpha (F_{\overrightarrow{V}}(w))^{-\frac{\alpha}{J_v(w)}} \\
&= (Z_h \cdot g^\beta)^\alpha (Z_h^{J_v(w)} \cdot (g^\beta)^{J_v(w)} \cdot g^{K_v(w)})^{-\frac{\alpha}{J_v(w)}} \\
&= Z_{h+1} \cdot Z_1^\beta \cdot Z_{h+1}^{-\frac{J_v(w)}{J_v(w)}} \cdot Z_1^{-\beta\frac{J_v(w)}{J_v(w)}} \cdot Z_1^{-\frac{K_v(w)}{J_v(w)}} \\
&= Z_1^{-\frac{K_v(w)}{J_v(w)}}
\end{aligned}$$

This means $Z_1^{-\frac{K_v(w)}{J_v(w)}}$ gives a token with the implicit random factor equals to $-\frac{\alpha}{J_v(w)}$. Randomization can be done easily by multiplying the above term by $(F_{\overrightarrow{V}}(w))^r$ where $r \in \mathbb{Z}_p^*$. Since $J_{u_1}(\mathsf{ID}_1) = \cdots = J_{u_k}(\mathsf{ID}_k) = 0 \bmod \rho_u$, all $\frac{\alpha}{J_v(w)}$ power terms appear in the construction of the token can be computed from Z_1.

User partial decryption: $\mathcal{A}$ queries $\mathcal{S}$'s oracle $\mathsf{DecO}^U(C, \mathsf{pk}, D)$. $\mathcal{S}$ performs the usual ciphertext validity checking to reject any invalid ciphertext C that is purported to be encrypted under $\overrightarrow{ID}$ and pk, and the token validity checking to reject any invalid token D that is purported to be a partial decryption of C. These validity checks prevent loss of information about the secret key sk. In particular, without the token checking, it is trivial for a Type-II adversary to derive the message in the challenge ciphertext by asking $\mathsf{DecO}^U(C^*, \mathsf{pk}^*, D')$ where D' is some invalid token derived from a valid one.

Suppose C is a valid ciphertext and D is valid for C and $\overrightarrow{ID}$, $\mathsf{DecO}^U(C, \mathsf{pk}, D)$ should give a correct decryption. For decrypting a valid ciphertext (C_1, C_2, τ, σ) with hash w, we have $\tau = g^s$ and $\sigma = F_{\overrightarrow{V}}(w)^s$ for some $s \in \mathbb{Z}_p^*$, i.e. $\sigma = g_2^{s \cdot J_v(w)} \cdot (g^s)^{K_v(w)}$. $\mathcal{S}$ can get g_2^s by $(\sigma / \tau^{K_v(w)})^{\frac{1}{J_v(w)}}$, $\hat{e}(Y, g_2)^s$ can thus be computed easily. Note that the secret key sk that matches pk is never explicitly used.

Complete decryption: If validity checking is passed, $\mathcal{S}$ returns

$$m = C_1 / \hat{e}(Y, (\sigma / \tau^{K_v(w)})^{\frac{1}{J_v(w)}}).$$

Game 7 (User Secret Key Extraction). If *mode* = II, $\mathcal{A}$ may issue UskO query on some public key $\mathsf{pk}_u \in \{\mathsf{pk}_1^*, \cdots \mathsf{pk}_{q_K}^*\}$. $\mathcal{S}$ picks a random integer q' from $\{1, \cdots, q_K\}$. If $\mathcal{A}$ issues the query $\mathsf{UskO}(\mathsf{pk}_{q'}^*)$, $\mathcal{S}$ aborts; returns θ_i for $i \neq q'$. This gives $\Pr[S_7] = \Pr[S_6]$ for *mode* = I and $\Pr[S_7] = \Pr[S_6]/q_K$ for *mode* = II.

Game 8 (Simulation of the Ciphertext / Embedding of the Problem Instance). Depending on whether the adversary is an insider or the server, we

have different modes of simulations. Now $\mathcal{S}$ introduces a variable $\gamma \in_R \mathbb{Z}_p^*$ and sets $\tau^* = g^\gamma$.

If $mode = \mathrm{I}$, g_1 is set to $Z_1 = g^\alpha$. $\mathcal{A}$ chooses an identifier $\overrightarrow{ID}^* = (\mathsf{ID}_1^*, \cdots, \mathsf{ID}_k^*)$, a public key $\mathsf{pk}^* = (X^*, Y^*)$ to be challenged with. $\mathcal{S}$ proceeds if $\hat{e}(Y^*, g) = \hat{e}(X^*, g_1)$. Let $T = (g^{\alpha^{h+1}})^\gamma$, $\mathcal{S}$ computes C_1^* by

$$\begin{aligned} & m_\iota \cdot \hat{e}(X^*, T) \cdot (Y^*, g^\gamma)^\beta \\ = & \, m_\iota \cdot \hat{e}(X^*, (g^{\alpha^{h+1}})^\gamma) \cdot (Y^*, g^\gamma)^\beta \\ = & \, m_\iota \cdot \hat{e}(Y^*, g^{\alpha^h})^\gamma \cdot (Y^*, g^\beta)^\gamma \\ = & \, m_\iota \cdot \hat{e}(Y^*, Z_h \cdot g^\beta)^\gamma \\ = & \, m_\iota \cdot \hat{e}(Y^*, g_2)^\gamma \end{aligned}$$

Note that it is the first time in the simulation that β is used directly (i.e. not in the form of g^β) except the computation of Msk, which is fine since $\mathcal{S}$ does not need to compute it explicitly for a Type-I adversary.

If $mode = \mathrm{II}$, $\mathcal{S}$ introduces a variable $\delta \in_R \mathbb{Z}_p^*$, and computes $(X_{q'}^*, Y_{q'}^*)$ by $((g^\delta)^{\theta_{q'}}, (g^\delta)^{\theta_{q'}\alpha})$ instead. Under the artificial abort in Game 7, $\mathcal{S}$ correctly guessed the public key $\mathcal{A}$ wants to attack. Let $T = g^{\beta\gamma\delta}$, $\mathcal{S}$ computes C_1^* by

$$\begin{aligned} & m_\iota \cdot (\hat{e}(g^\delta, g^\gamma)^{\alpha^{h+1}} \cdot \hat{e}(g^\alpha, T))^{\theta_i} \\ = & \, m_\iota \cdot (\hat{e}(g^\delta, g^{\alpha^{h+1}})^\gamma \cdot \hat{e}(g^\alpha, g^{\beta\gamma\delta}))^{\theta_i} \\ = & \, m_\iota \cdot \hat{e}(g^{\delta\theta_i\alpha}, g^{\alpha^h})^\gamma \cdot \hat{e}(g^{\delta\theta_i\alpha}, g^\beta)^\gamma \\ = & \, m_\iota \cdot \hat{e}(Y^*, Z_h \cdot g^\beta)^\gamma \\ = & \, m_\iota \cdot \hat{e}(Y^*, g_2)^\gamma \end{aligned}$$

Note that it is the first time in the simulation that α is used directly (i.e. not in the form of $g^\alpha, \cdots, g^{\alpha^h}$).

In both modes, $\mathcal{S}$ sets $C_2^* = \prod_{j=1}^k (g^\gamma)^{K_{u_j}(\mathsf{ID}_j^*)}$, $\sigma^* = (g^\gamma)^{K_v(w^*)}$ where $w^* = H(C_1^*, C_2^*, \tau^*, \overrightarrow{ID}^*, \mathsf{pk}^*)$ for the rest of the challenge, which is a perfect simulation if $\mathcal{S}$ did not abort in Game 4. We have $\Pr[S_8] = \Pr[S_7]$.

Game 9 (The Indistinguishability Cards). If $mode = \mathrm{I}$, $\mathcal{S}$ forgets (α, γ). If $mode = \mathrm{II}$, $\mathcal{S}$ forgets (β, γ, δ). $\mathcal{S}$ can simulate the game in both modes as long as $(g^\alpha, \cdots, g^{\alpha^h}, g^\gamma)$ are known for $mode = \mathrm{I}$ or $(g^\beta, g^\gamma, g^\delta)$ are known for $mode = \mathrm{II}$, except computing the term T. Now $\mathcal{S}$ just picks a $T \in_R \mathbb{G}$. The transition from Game 8 to Game 9 is based on the intractability of either h-wDHI or 3-DDH. Both games are equal unless there exists a PPT algorithm $\mathcal{D}$ that distinguishes T from random. Therefore, we have $|\Pr[S_9] - \Pr[S_8]| \leq Adv_{\mathcal{D}}^{\mathrm{X}}(\lambda)$ where X is either h-wDHI or 3-DDH. Finally, C_1^* perfectly hides m_ι from $\mathcal{A}$, we have $\Pr[S_9] = 1/2$.

B Security Models of Timed-Release Encryption

We following the typical TRE formulation [3, 8, 12, 19, 24] such that public key is certified and the typical TRE security model [8, 12, 19, 24] such that only a

single public key is considered. However, we extend the existing notion such that partial decryption by a security-mediator is supported.

We consider the two kinds of adversaries. A Type-I adversary models any coalition of rogue users, and who aims to break the confidentiality of another user's ciphertext. A Type-II adversary that models a curious time server, who aims to break the confidentiality of a user's ciphertext. Security against these adversaries are modeled by the experiment below for $\mathsf{X} \in \{\mathrm{I}, \mathrm{II}\}$, denoting whether an PPT adversary $\mathcal{A} = (\mathcal{A}_{\mathsf{find}}, \mathcal{A}_{\mathsf{guess}})$ is of Type-I or Type-II. The allowed oracle queries $\mathcal{O}$ and the auxiliary information Aux depends on X.

Definition 9. Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA'-X}}(\lambda)$

$(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}^*, \mathsf{sk}^*) \xleftarrow{\$} \mathsf{KGen}$
$(m_0, m_1, \mathsf{ID}^*, \mathsf{state}) \xleftarrow{\$} \mathcal{A}_{\mathsf{find}}^{\mathcal{O}}(\mathsf{Pub}, \mathsf{Aux}, \mathsf{pk}^*)$
$b \xleftarrow{\$} \{0,1\}$, $C^* \xleftarrow{\$} \mathsf{Enc}(m_b, \mathsf{ID}^*, \mathsf{pk}^*)$
$b' \xleftarrow{\$} \mathcal{A}_{\mathsf{guess}}^{\mathcal{O}}(C^*, \mathsf{state})$
If $(|m_0| \neq |m_1|) \vee (b \neq b')$ *then return 0 else return 1*

$\mathcal{O}$ *is a set of oracles* $\mathsf{ExtractO}(\cdot), \mathsf{DecO}^S(\cdot,\cdot), \mathsf{DecO}^U(\cdot,\cdot,\cdot), \mathsf{DecO}(\cdot,\cdot,\cdot)$ *as below.*

1. *An* $\mathsf{ExtractO}$ *oracle that takes an identifier* $\mathsf{ID} \in \{0,1\}^n$ *as input and returns its trapdoor* d_{ID}.
2. *A* DecO^S *oracle that takes a ciphertext* C *and an identifier* ID*, and outputs* $\mathsf{Dec}^S(C, d_{\mathsf{ID}})$. *Note that* C *may or may not be encrypted under* ID.
3. *A* DecO^U *oracle that takes a ciphertext* C*, a public key* pk *and a token* D*, and outputs* $\mathsf{Dec}^U(C, \mathsf{sk}, D)$ *where* sk *is the secret key that matches* pk.
4. *A* DecO *oracle that takes a ciphertext* C*, an identifier* ID*, and a public key* pk*, and outputs* $\mathsf{Dec}^U(C, \mathsf{sk}, D)$ *where* sk *is the secret key that matches* pk *and* $D = \mathsf{Dec}^S(C, d_{\mathsf{ID}})$. *Note that* C *may not be encrypted under* ID *and* pk.

Definition 10. *A timed-release encryption scheme is* (t, q_E, q_D, ϵ) *CCA-secure against a Type-I adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA'-I}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_E *extraction queries and* q_D *decryption queries (of any type), subjects to the following constraints:*

1. $\mathsf{Aux} = (\mathsf{sk}^*)$*, i.e. the user secret key is given to the adversary.*
2. *No* $\mathsf{ExtractO}(\mathsf{ID}^*)$ *query throughout the game.*
3. *No* $\mathsf{DecO}^S(C^*, \mathsf{ID}^*)$ *query throughout the game.*
4. *No* $\mathsf{DecO}(C^*, \mathsf{ID}^*, \mathsf{pk}^*)$ *query throughout the game.*

Definition 11. *A timed-release encryption scheme is* (t, q_D, ϵ) *CCA-secure against a Type-II adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA'-II}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_D *decryption queries, under the following conditions:*

1. $\mathsf{Aux} = (\mathsf{Msk})$*, i.e. the master secret key is given to the adversary.*
2. *No* $\mathsf{DecO}^U(C^*, \mathsf{pk}^*, D)$ *query throughout the game, where* D *is outputted by the algorithm* $\mathsf{Dec}^S(C^*, d_{\mathsf{ID}}^*)$.
3. *No* $\mathsf{DecO}(C^*, \mathsf{ID}^*, \mathsf{pk}^*)$ *query throughout the game.*