IEICE TRANS. FUNDAMENTALS, VOL.E94–A, NO.8 AUGUST 2011

PAPER

A Timed-Release Proxy Re-Encryption Scheme*

Keita EMURA[†a)], **Atsuko MIYAJI**[††], ***Members***, ***and*** **Kazumasa OMOTE**[††], ***Nonmember***

SUMMARY Timed-Release Encryption (TRE) is a kind of time-dependent encryption, where the time of decryption can be controlled. More precisely, TRE prevents even a legitimate recipient decrypting a ciphertext before a semi-trusted Time Server (TS) sends trapdoor s_T assigned with a release time T of the encryptor's choice. Cathalo et al. (ICICS2005) and Chalkias et al. (ESORICS2007) have already considered encrypting a message intended for multiple recipients with the same release time. One drawback of these schemes is the ciphertext size and computational complexity, which depend on the number of recipients N. Ideally, it is desirable that any factor (ciphertext size, computational complexity of encryption/decryption, and public/secret key size) does not depend on N. In this paper, to achieve TRE with such fully constant costs from the encryptor's/decryptor's point of view, by borrowing the technique of Proxy Re-Encryption (PRE), we propose a cryptosystem in which even if the proxy transformation is applied to a TRE ciphertext, the release time is still effective. By sending a TRE ciphertext to the proxy, an encryptor can foist N-dependent computation costs on the proxy. We call this cryptosystem Timed-Release PRE (TR-PRE). This function can be applied to efficient multicast communication with a release time indication.

key words: *timed-release encryption, proxy re-encryption*

1. Introduction

Timed-Release Encryption (TRE) was proposed by May [33], and is a kind of time-dependent encryption, where the time of decryption can be controlled. Even a legitimate recipient cannot decrypt a ciphertext before a semi-trusted Time Server (TS) sends (or broadcasts) trapdoor s_T assigned with release time T of the encryptor's choice. Similarly, TS (which can generate all trapdoors) cannot decrypt a ciphertext, since TS does not have the legitimate decryption key. That is, both s_T and the legitimate recipient's decryption key are indispensable to decrypt a ciphertext assigned with T. To guarantee this, TS is modeled as a honest-but-curious adversary in the security model (called *time server security* [22], [34]). Such adversary follows the protocol correctly, but may try to obtain additional information.

Although usual TREs deal with only a single recipient, Cathalo et al. [13] and Chalkias et al. [14] have already considered encrypting a message intended for several recipients with the same release time. Schemes by Cathalo et al. and Chalkias et al. are efficient compared with previous TRE schemes with recipient-to-recipient encryption, since the most costly part (especially pairing computation $e(\cdot,\cdot)$ or group operation on the range of the pairing, e.g., $e(\cdot,\cdot)^r$ for some exponent r) has only to be computed once, and this element is used commonly. Note that, since Chow et al. [18] showed that the Chalkias et al. scheme [14] is vulnerable under the CCA attack, we pay attention to the Cathalo et al. scheme in the following discussion. Informally, for a common release time T and number of recipients N, the form of a ciphertext in the Cathalo et al. scheme is: $(C_1, C_2, \ldots, C_N, (M\|\text{random nonce}) \oplus K), \mathit{RecipientList}, T)$, where $K = \mathit{Hash}(e(\cdot,\cdot))$ is a commonly used ephemeral key computed by both C_i and a user U_i's secret key. One drawback of this scheme is the ciphertext size, namely, the length of the ciphertext depends on N (See Fig. 1). As a simple countermeasure, if each ciphertext (for a user U_i) is represented as $(C_i, (M\|\text{random nonce}) \oplus K, T)$, then actual transferred ciphertext size is constant. Nevertheless, there is still a remaining problem, where the encryption cost also depends on N. This can be a serious problem when N becomes large. Ideally, it is desirable that any factor (ciphertext size, computational complexity of encryption/decryption, and public/secret key size) does not depend on N.

Due to the fact that typical group-oriented encryption systems (e.g., broadcast encryption [21], hierarchical identity-based encryption (IBE) [8], and others) only satisfy partially constant costs (i.e., at least one of the factor de-

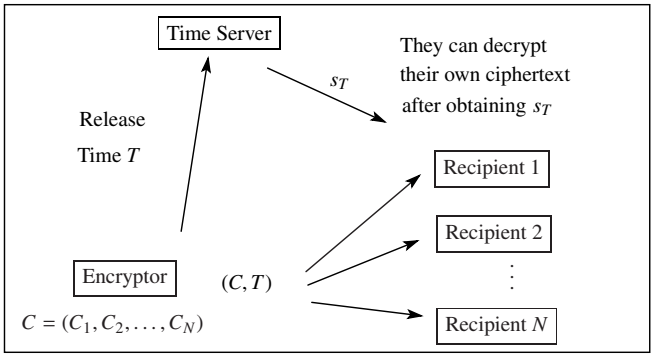

Fig. 1 Previous TRE for multiple recipients [24].

Manuscript received November 29, 2010.
Manuscript revised March 24, 2011.

[†]The author is with the Center for Highly Dependable Embedded Systems Technology, Japan Advanced Institute of Science and Technology (JAIST), Nomi-shi, 923-1292 Japan.
[††]The authors are with the School of Information Science, JAIST, 923-1292 Japan.

*A preliminary version of this paper appears in the 4th International Conference on Provable Security, ProvSec 2010 [24]. This is the full version.

a) E-mail: k-emura@jaist.ac.jp

DOI: 10.1587/transfun.E94.A.1682

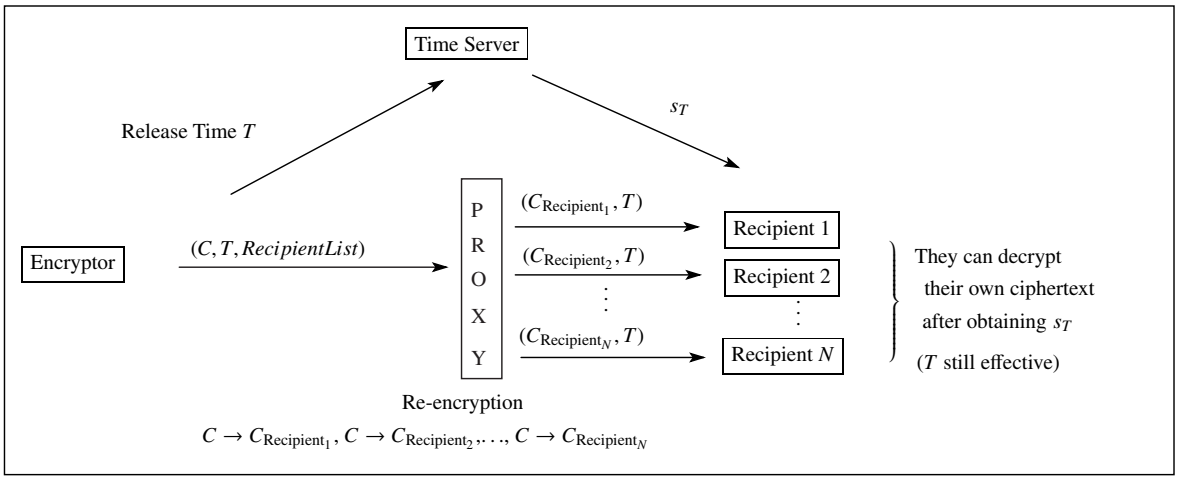

Fig. 2 Our TR-PRE with multiple recipients [24].

pends on N), it seems hard to directly[†] construct TRE with such fully constant costs. Therefore, we need to investigate another approach to achieve TRE with the fully constant costs.

As a solution, we focus on that TRE with the fully constant costs *from the encryptor's/decryptor's point of view* is highly significant from the perspective of usability. So, here we consider an additional agency who takes over N-dependent computation costs. Of course, if such additional agency is fully trusted, then we can easily achieve such TRE system. For example, an encryptor computes a ciphertext by using the agency's public key, and sends it with the recipient list and the release time T to the agency. The agency decrypts the ciphertext (and therefore the agency can know all plaintexts), re-encrypts the resulting plaintext by using the corresponding recipient's public key and the release time T, and sends TRE ciphertexts to each recipient. In this solution, the agency has a special privilege that the agency can know all plaintexts. However, in the conventional TRE, TS is modeled as the semi-trusted agency for ensuring that only a legitimate recipient can decrypt a ciphertext encrypted by the recipient's public key. That is, the additional agency should be modeled as semi trusted (as in TS). So, we need to investigate a methodology how to foist N-dependent computation costs to the agency ensuring that only a legitimate recipient can decrypt a ciphertext.

Proxy Re-Encryption (PRE) [5] is a candidate cryptographic primitive to implement a semi-trusted agency who takes over N-dependent computation costs. Briefly, a semi-trusted intermediate agency, called *proxy*, transforms a ciphertext made by a delegator's public key into a re-encrypted ciphertext that can be decrypted using a delegatee's secret key. For example, the proxy can forward a ciphertext for a delegatee (or potentially plural delegatees) without decrypting the ciphertext. This functionality seems applicable to TRE for multiple recipients, however, any methodology to apply PRE for reducing costs of TRE has not been proposed so far.

Our contribution: In this paper, to achieve TRE with fully constant costs from the encryptor's/decryptor's point of view, by borrowing the technique of PRE, we propose a cryptosystem in which even if the proxy transformation is applied to a TRE ciphertext, the release time is still effective. By sending a TRE ciphertext to the proxy, an encryptor can foist N-dependent computation costs on the proxy. We call this cryptosystem Timed-Release PRE (TR-PRE)[††]. Informally, the flow of TR-PRE is as follows (illustrated in Fig. 2). An encryptor computes a TRE ciphertext under the particular public key, and the proxy translates this ciphertext into re-encrypted ciphertexts under each recipient. The proxy sends the corresponding re-encrypted ciphertext to each recipient. Each recipient can decrypt it after the corresponding trapdoor s_T is released.

As in TRE, the condition that no authority can decrypt ciphertexts should be satisfied. To do so, the proxy is modeled as a semi-trusted agency, and we assume that not only the proxy follows the protocol but also the proxy will not collude with receivers as in [19], [27], since the proxy and a receiver can decrypt any ciphertext by colluding with each other.

By applying PRE functionality, an encryptor can transfer these N-dependent parts to the proxy, and therefore the number of ciphertexts (and computational complexity of encryption/decryption also) does not depend on the number of recipients N. In addition, our TR-PRE achieves constant public/secret key size. The factor depending on N is the proxy re-encryption costs only. So, TR-PRE can work like TRE with fully constant costs from the encryptor's and decryptor's point of view. One trade-off of this efficiency, information that who recipients are is known by the proxy as in PRE.

[†]The words "directly construction" mean construction in the conventional TRE framework, namely, trying to construct TRE with fully constant costs without adding any functionality to the original TRE functionalities such as Cathalo et al. scheme.

[††]Note that Attribute-Based PRE (AB-PRE) [31] is not suitable for constructing TRE scheme with fully constant costs even if the release time is regarded as an attribute (we explain it in the Sect. 5.3).

Organization: The paper is organized as follows: Security definitions of TR-PRE are presented in Sect. 3. Our scheme is described in Sect. 4. The security analyses are presented in Sect. 5. Efficiency comparisons and applications of TR-PRE are presented in Sect. 6.

2. Preliminaries

In this section, we give the definitions of bilinear groups and complexity assumptions which are applied in our TR-PRE construction. In the following descriptions, $x \xleftarrow{\$} S$ means that x is chosen uniformly from a set S. $y \leftarrow A(x)$ means that y is an output of an algorithm A under an input x.

2.1 Bilinear Groups and Complexity Assumptions

Definition 1. (**Bilinear Groups**) *Bilinear groups and a bilinear map are defined as follows:*

1. $\mathbb{G}$ *and* $\mathbb{G}_T$ *are cyclic groups of prime order* p.
2. g *is a generator of* $\mathbb{G}$.
3. e *is an efficiently computable bilinear map* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *with the following properties.*
   - *Bilinearity: for all* $u, u', v, v' \in \mathbb{G}$, $e(uu', v) = e(u, v)e(u', v)$ *and* $e(u, vv') = e(u, v)e(u, v')$.
   - *Non-degeneracy:* $e(g, g) \neq 1_{\mathbb{G}_T}$ ($1_{\mathbb{G}_T}$ *is the* $\mathbb{G}_T$ *unit*).

Definition 2 (3-QDBDH assumption [32])**.** *The 3-Quotient Decision Bilinear Diffie-Hellman (3-QDBDH) problem is a problem in which, for input of a tuple* $(g, g^a, g^{a^2}, g^{a^3}, g^b, Z) \in \mathbb{G}^5 \times \mathbb{G}_T$, *to decide whether* $Z = e(g, g)^{b/a}$ *or not. We say that the 3-QDBDH assumption holds in* $\mathbb{G}$ *if the advantage* $Adv^{3\text{-}QDBDH}_{\mathbb{G},\mathcal{A}}(1^k) := |\Pr[\mathcal{A}(g, g^a, g^{a^2}, g^{a^3}, g^b, e(g, g)^{b/a}) = 0] - \Pr[\mathcal{A}(g, g^a, g^{a^2}, g^{a^3}, g^b, e(g, g)^z) = 0]|$, *where* $e(g, g)^z \in \mathbb{G}_T \setminus \{e(g, g)^{b/a}\}$, *is negligible for any probabilistic polynomial-time (PPT) algorithm* $\mathcal{A}$.

The hardness of the 3-QDBDH problem was discussed in [32], where the 3-QDBDH problem is not easier than the q-Decisional Bilinear Diffie-Hellman Inversion (q-DBDHI) problem [7]. The difficulty of the q-DBDHI problem in generic groups was shown in [23], and this result implies the difficulty of the 3-QDBDH problem in generic groups. As in [32], we use the modified version of the 3-QDBDH (modified 3-QDBDH) problem, where for input of a tuple $(g, g^{1/a}, g^a, g^{a^2}, g^b, Z) \in \mathbb{G}^5 \times \mathbb{G}_T$, to decide whether $Z = e(g, g)^{b/a^2}$ or not. This modified 3-QDBDH problem is equivalent to the 3-QDBDH problem (See [32] Lemma 1).

Definition 3 (Truncated decisional q-ABDHE assumption [26])**.** *The truncated decisional q-Augmented Bilinear Diffie-Hellman (q-ABDHE) problem is a problem in which, for input of a tuple* $(g', g'^{(\alpha^{q+2})}, g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^q}, Z) \in \mathbb{G}^{q+3} \times \mathbb{G}_T$, *to decide whether* $Z = e(g, g')^{\alpha^{q+1}}$ *or not. We say that the truncated decisional q-ABDHE assumption holds in* $\mathbb{G}$ *if the advantage* $Adv^{q\text{-}ABDHE}_{\mathbb{G},\mathcal{A}}(1^k) := |\Pr[\mathcal{A}(g', g'^{(\alpha^{q+2})}, g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^q}, e(g, g')^{\alpha^{q+1}}) = 0] - \Pr[\mathcal{A}(g', g'^{(\alpha^{q+2})}, g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^q}, e(g, g')^z) = 0]|$, *where* $e(g, g')^z \in \mathbb{G}_T \setminus \{e(g, g')^{\alpha^{q+1}}\}$, *is negligible for any PPT algorithm* $\mathcal{A}$.

2.2 Strongly Existential Unforgeable One-Time Signatures

We apply the Libert-Vergnaud PRE [32], which needs the CHK transformation [11] to satisfy CCA security by strongly existential unforgeable (sUF) one-time signatures (e.g., [3]). So, here we introduce sUF one-time signature as follows. An sUF one-time signature consists of three algorithms, Sig.KeyGen, Sign and Verify. Sig.KeyGen is a probabilistic algorithm which outputs a signing/verification key pair (K_s, K_v). Sign is a probabilistic algorithm which outputs a signature σ from K_s, and a message $M \in \mathcal{M}_{Sig}$, where $\mathcal{M}_{Sig}$ is the message space of a signature scheme. Verify is a deterministic algorithm which outputs a bit from σ, K_v and M. "Verify outputs 1" indicates that σ is a valid signature of M, and 0, otherwise. The security experiment of sUF one-time signature under an adaptive Chosen Message Attack (one-time sUF-CMA) is defined as follows:

Definition 4 (one-time sUF-CMA)**.** *We say that a signature scheme is one-time sUF-CMA secure if the advantage* $Adv^{one\text{-}time\ sUF\text{-}CMA}_{\mathcal{A}}(1^k)$ *is negligible for any polynomial-time adversary* $\mathcal{A}$ *in the following experiment.*

$$
\begin{aligned}
&Adv^{one\text{-}time\ sUF\text{-}CMA}_{\mathcal{A}}(1^k) = \\
&\quad \Pr\left[(K_s, K_v) \leftarrow \mathsf{Sig.KeyGen}(1^k);\right. \\
&\quad (M, State) \leftarrow \mathcal{A}(K_v); \sigma \leftarrow \mathsf{Sign}(K_s, M); \\
&\quad \left.(M^*, \sigma^*) \leftarrow \mathcal{A}(K_v, \sigma, State); \right. \\
&\quad \left.(M^*, \sigma^*) \neq (M, \sigma); \mathsf{Verify}(K_v, \sigma^*, M^*) = 1\right]
\end{aligned}
$$

3. Definitions of TR-PRE

3.1 Functions of (Single-Hop) TR-PRE

First, we introduce encryption levels (refer to [1]) for single-hop PRE as follows: A ciphertext computed by the Encrypt2 algorithm is called the "second-level" ciphertext, which can be re-encrypted using an appropriate re-encryption key. A ciphertext computed by the Re-Encrypt algorithm or the Encrypt1 algorithm is called the "first-level" ciphertext, which cannot be re-encrypted for any user. A ciphertext is identified whether it is the first one or not, since the form of the first and second ciphertext is different in our TR-PRE (and the Libert-Vergnaud PRE also). A TR-PRE scheme Π consists of eight algorithms (Setup, KeyGen, Encrypt1, Encrypt2, TS-Release, RK-Gen, Re-Encrypt, Decrypt):

Definition 5. *TR-PRE*

$\mathsf{Setup}(1^k)$ *: This algorithm takes as input the security parameter k, and returns the master public parameters params, the TS's public key* TS_{pub}*, and the TS's secret key* ts_{priv}*. We assume that params includes* TS_{pub}*.*

$\mathsf{KeyGen}(params)$ *: This algorithm takes as input params, and returns a public/secret key pair* (upk, usk)*.*

$\mathsf{TS\text{-}Release}(params, ts_{\mathsf{priv}}, T)$ *: This algorithm takes as input params,* ts_{priv}*, and a release time T, and returns a trapdoor* s_T*.*

$\mathsf{Encrypt1}(params, upk, M, T)$ *: This algorithm takes as input params, a user's public key upk, a plaintext M, and T, and returns a first-level ciphertext C which cannot be transformed.*

$\mathsf{Encrypt2}(params, upk, M, T)$ *: This algorithm takes as input params, a user's public key upk, a plaintext M, and T, and returns a second-level ciphertext C which can be transformed into the first-level ciphertext using an appropriate re-encryption key.*

$\mathsf{RKGen}(params, usk_i, upk_j)$ *: This algorithm takes as input params, a user* U_i*'s secret key* usk_i*, and a user* U_j*'s public key* upk_j*, and returns a re-encryption key* R_{ij}*.*

$\mathsf{Re\text{-}Encrypt}(params, R_{ij}, upk_i, C)$ *: This algorithm takes as input params,* R_{ij}*, and* upk_i*, and a second-level ciphertext C encrypted by* upk_i*, and returns the first-level re-encrypted ciphertext C which can be decrypted by* usk_j*.*

$\mathsf{Decrypt}(params, usk, s_T, C, T)$ *: This algorithm takes as input params, usk,* s_T *and C, and returns M or* $\perp$*.*

3.2 A Typical Usage of TR-PRE

Here, we describe a typical usage of TR-PRE.

Setup Phase: We assume that each user executes the KeyGen algorithm, and obtains its own key pair. Let (upk, usk) be the key pair of the encryptor of the following explanation, and *RecipientList* be the set of recipient. The encryptor computes re-encryption keys $R_j \leftarrow \mathsf{RKGen}(params, usk, upk_j)$ for all $U_j \in$ *RecipientList*. This procedure can be done before the actual encryption procedure. In addition, once a re-encryption key is stored in the proxy, this key can be continually used after that. Therefore, we assume that the computation of re-encryption keys has already been done before the actual encryption.

Encryption Phase: An encryptor computes a TRE ciphertext *by using its own public key upk*, and sends (C, T) with (recipient list) to the proxy. The proxy re-encrypts (C, T) by using its re-encryption key, and sends the re-encryption result to the corresponding recipient.

Of course, we can assume that an encryptor computes a TRE ciphertext by using a recipient (say U_i) public key. In this case, however, we need to assume that re-encryption keys from the U_i to $U_j \in$ *RecipientList* has already been preserved in the proxy. Since the encryptor decides *RecipientList*, our scenario (the encryptor uses its own public key) is reasonable in practice.

3.3 Security Requirements

First, we define the correctness of TR-PRE as follows. Correctness guarantees that the honestly computed ciphertext and the honestly re-encrypted ciphertext can be correctly decrypted by using the appropriate secret key and the appropriate trapdoor.

Definition 6 (Correctness)**.** *For all* $(params, ts_{priv}) \leftarrow \mathsf{Setup}(1^k)$*,* $(upk_i, usk_i), (upk_j, usk_j) \leftarrow \mathsf{KeyGen}(params)$*, T, M, and* $s_T \leftarrow \mathsf{TS\text{-}Release}(params, ts_{priv}, T)$*,*

$$M = \mathsf{Decrypt}\Big(params, usk_i, s_T, \mathsf{Encrypt2}(params, upk_i, M, T), T\Big),$$

$$M = \mathsf{Decrypt}\Big(params, usk_i, s_T, \mathsf{Encrypt1}(params, upk_i, M, T), T\Big), \text{ and}$$

$$M = \mathsf{Decrypt}\Big(params, usk_j, s_T, \mathsf{Re\text{-}Encrypt}(params, \mathsf{RKGen}(params, usk_i, upk_j), upk_i, \mathsf{Encrypt2}(params, upk_i, M, T)), T\Big)$$

hold.

Next, we define the chosen-ciphertext security requirements of TR-PRE. These are naturally defined from the security definitions of the Cathalo et al. TRE [13] and the Libert-Vergnaud PRE [32].

First, we define *replayable chosen-ciphertext* (IND-RCCA) security. IND-RCCA security guarantees that even if the appropriate trapdoor is given, non-legitimate users (who do not have an appropriate secret key) cannot decrypt a ciphertext. This suggests $\mathcal{A}$ is an "honest but curious" TS. We give two IND-RCCA security notions at second-level ciphertext and first-level ciphertext, respectively. In the following experiments, for the challenge public/secret key, ciphertext, and plaintext, these are superscripted by $*$. For honest parties, keys are subscripted by h or h'. For corrupted parties, keys are subscripted by c or c'.

First, we define IND-RCCA security at second-level ciphertext. As in [32], a PPT adversary $\mathcal{A}$ is given all re-encryption keys, except from the target user to a corrupted user. As in [12], [32], we assume a static corruption model, which does not capture a scenario in which an adversary generates public/secret keys for all parties.

Oracles: $\mathcal{A}$ can access the re-encryption oracle $\mathcal{O}_{RE\text{-}ENC}$ and the decryption oracle $\mathcal{O}_{DEC}$ which are defined as follows. For an input (upk_i, upk_j, C), $\mathcal{O}_{RE\text{-}ENC}$ returns $\perp$ if the one of following holds: (1) C is the first-level ciphertext, or (2) upk_j is a corrupted user and $(upk_i, C) = (upk^*, C^*)$, or (3) C is not properly computed by using upk_i, or (4) either upk_i or upk_j were not generated the KeyGen algorithm executed by the challenger. Otherwise, $\mathcal{O}_{RE\text{-}ENC}$ returns a re-encrypted ciphertext $C' =$

$\mathsf{Re\text{-}Encrypt}(params, \mathsf{RKGen}(params, usk_i, upk_j), upk_i, C)$. For an input (upk, C, T), O_{DEC} returns $\perp$ if the one of following holds: (1) upk was not produced by the KeyGen algorithm executed by the challenger, or (2) $(upk, C, T) = (upk^*, C^*, T^*)$, or (3) (upk, C) is a derivative of (upk^*, C^*), where we say that (upk, C) is a *derivative* of (upk^*, C^*) if $\mathsf{Decrypt}(params, usk, s_T, C, T) \in \{M_0^*, M_1^*\}$ for any (queried) T whatever $T \neq T^*$, C is a first level ciphertext and either $upk = upk^*$ or $upk \in \{upk_h\}$. Otherwise, O_{DEC} returns a decryption result M.

Definition 7 (IND-RCCA Security at Second-level Ciphertext)**.** *We say that a (single-hop) TR-PRE scheme is IND-RCCA secure at second-level ciphertext if the advantage $Adv_{\Pi,\mathcal{A}}^{IND\text{-}RCCA\text{-}2nd}(1^k)$ is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
&Adv_{\Pi,\mathcal{A}}^{IND\text{-}RCCA\text{-}2nd}(1^k) = \\
&\quad \Big| \Pr\big[(params, ts_{priv}) \leftarrow \mathsf{Setup}(1^k); \\
&\quad (upk^*, usk^*) \leftarrow \mathsf{KeyGen}(params); \\
&\quad \{(upk_h, usk_h) \leftarrow \mathsf{KeyGen}(params)\}; \\
&\{(upk_c, usk_c) \leftarrow \mathsf{KeyGen}(params)\}; \\
&Set\ Keys := \{upk^*, \{upk_h\}, \{(upk_c, usk_c)\}\}; \\
&\{R_{c*} \leftarrow \mathsf{RKGen}(params, usk_c, upk^*)\}; \\
&\{R_{h*} \leftarrow \mathsf{RKGen}(params, usk_h, upk^*)\}; \\
&\{R_{*h} \leftarrow \mathsf{RKGen}(params, usk^*, upk_h)\}; \\
&\{R_{hc} \leftarrow \mathsf{RKGen}(params, usk_h, upk_c)\}; \\
&\{R_{ch} \leftarrow \mathsf{RKGen}(params, usk_c, upk_h)\}; \\
&\{R_{cc'} \leftarrow \mathsf{RKGen}(params, usk_c, upk_{c'})\}; \\
&\{R_{hh'} \leftarrow \mathsf{RKGen}(params, usk_h, upk_{h'})\}; \\
&Set\ ReKeys := \{\{R_{c*}\}, \{R_{h*}\}, \{R_{*h}\}, \{R_{hc}\}, \\
&\qquad \{R_{ch}\}, \{R_{cc'}\}, \{R_{hh'}\}\}; \\
&(M_0^*, M_1^*, T^*, State) \\
&\quad \leftarrow \mathcal{A}^{O_{RE\text{-}ENC}, O_{DEC}}(params, Keys, ReKeys, ts_{priv}); \\
&\mu \overset{\$}{\leftarrow} \{0,1\}; C^* \leftarrow \mathsf{Encrypt2}(params, upk^*, M_\mu^*, T^*); \\
&\mu' \leftarrow \mathcal{A}^{O_{RE\text{-}ENC}, O_{DEC}}(C^*, State); \mu = \mu'\big] - 1/2\Big|
\end{aligned}
$$

A ciphertext encrypted under upk from $\{upk_h\}$ can be re-encrypted for corrupt users by $\{R_{hc}\}$. In addition, second-level ciphertexts under upk^* can be translated for honest users by $\{R_{*h}\}$. Since the resulting ciphertexts can be queried for O_{DEC}, a second-level decryption oracle is useless.

Next, we define the IND-RCCA security at first-level ciphertext as follows. Since first-level ciphertexts cannot be re-encrypted, all re-encryption keys are given to $\mathcal{A}$. So, $O_{RE\text{-}ENC}$ is useless and is deleted. For the same reason, a second-level decryption oracle is also useless. The definition of O_{DEC} is the same as that of the second level one, except we say that (upk, C) is a *derivative* of (upk^*, C^*) if $\mathsf{Decrypt}(params, usk, s_T, C, T) \in \{M_0^*, M_1^*\}$ for any T and C is a first level ciphertext and $upk = upk^*$.

Definition 8 (IND-RCCA Security at First-level Ciphertext)**.** *We say that a (single-hop) TR-PRE scheme is IND-RCCA secure at first-level ciphertext if the advantage $Adv_{\Pi,\mathcal{A}}^{IND\text{-}RCCA\text{-}1st}(1^k)$ is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
&Adv_{\Pi,\mathcal{A}}^{IND\text{-}RCCA\text{-}1st}(1^k) = \\
&\quad \Big| \Pr\big[(params, ts_{priv}) \leftarrow \mathsf{Setup}(1^k); \\
&\quad (upk^*, usk^*) \leftarrow \mathsf{KeyGen}(params); \\
&\quad \{(upk_h, usk_h) \leftarrow \mathsf{KeyGen}(params)\}; \\
&\quad \{(upk_c, usk_c) \leftarrow \mathsf{KeyGen}(params)\}; \\
&\quad Set\ Keys := \{upk^*, \{upk_h\}, \{(upk_c, usk_c)\}\}; \\
&\quad \{R_{c*} \leftarrow \mathsf{RKGen}(params, usk_c, upk^*)\}; \\
&\quad \{R_{h*} \leftarrow \mathsf{RKGen}(params, usk_h, upk^*)\}; \\
&\quad \{R_{*h} \leftarrow \mathsf{RKGen}(params, usk^*, upk_h)\}; \\
&\quad \{R_{*c} \leftarrow \mathsf{RKGen}(params, usk^*, upk_c)\}; \\
&\quad \{R_{hc} \leftarrow \mathsf{RKGen}(params, usk_h, upk_c)\}; \\
&\{R_{ch} \leftarrow \mathsf{RKGen}(params, usk_c, upk_h)\}; \\
&\{R_{cc'} \leftarrow \mathsf{RKGen}(params, usk_c, upk_{c'})\}; \\
&\{R_{hh'} \leftarrow \mathsf{RKGen}(params, usk_h, upk_{h'})\}; \\
&Set\ ReKeys := \{\{R_{c*}\}, \{R_{h*}\}, \{R_{*h}\}, \{R_{*c}\}, \\
&\qquad \{R_{hc}\}, \{R_{ch}\}, \{R_{cc'}\}, \{R_{hh'}\}\}; \\
&(M_0^*, M_1^*, T^*, State) \\
&\quad \leftarrow \mathcal{A}^{O_{DEC}}(params, Keys, ReKeys, ts_{priv}); \\
&\mu \overset{\$}{\leftarrow} \{0,1\}; C^* \leftarrow \mathsf{Encrypt1}(params, upk^*, M_\mu^*, T^*); \\
&\mu' \leftarrow \mathcal{A}^{O_{RE\text{-}ENC}, O_{DEC}}(C^*, State); \mu = \mu'\big] - 1/2\Big|
\end{aligned}
$$

Next, we define *weak chosen-time period chosen-ciphertext* (IND-wCTCA) security[†]. IND-wCTCA security guarantees that even if $\mathcal{A}$ has the appropriate secret key, $\mathcal{A}$ cannot decrypt a ciphertext before the appropriate trapdoor is released. This suggests $\mathcal{A}$ is a malicious user in this experiment. As in the IND-RCCA security definitions, we give two IND-wCTCA security notions at second-level ciphertext and first-level ciphertext, respectively.

Oracles: $\mathcal{A}$ can access the key generation oracle O_{KeyGen}, the re-encryption oracle $O_{RE\text{-}ENC}$, the re-encryption key generation oracle O_{RKGen}, the timed-release trapdoor extraction oracle $O_{TS\text{-}Release}$, and the decryption oracle O_{DEC} which are defined as follows. O_{KeyGen} returns $(upk, usk) \leftarrow \mathsf{KeyGen}(params)$. For an input (upk_i, upk_j, C), $O_{RE\text{-}ENC}$ returns $\perp$ if the one of following holds: (1) C is the first-level ciphertext, (2) C is not properly computed by using upk_i, or (3) either upk_i or upk_j were not generated

[†]The notion "weak" means that our definition is weaker than the IND-CTCA security (which is defined in the TRE context) given by Cathalo et al. [13]. That is, when $\mathcal{A}$ generates upk and inputs (upk, C, T) to O_{DEC}, O_{DEC} has to answer without knowing the corresponding decryption key in the IND-CTCA sense, whereas O_{DEC} returns $\perp$ in the IND-wCTCA sense, since we assume a static corruption model.

the KeyGen algorithm executed by the challenger. Otherwise, $O_{RE\text{-}ENC}$ returns a re-encrypted ciphertext $C' = \mathsf{Re\text{-}Encrypt}(params, \mathsf{RKGen}(params, usk_i, upk_j), upk_i, C)$. For an input (usk_i, upk_j), O_{RKGen} returns $\perp$ if either usk_i or upk_j were not generated the KeyGen algorithm executed by the challenger. Otherwise, O_{RKGen} returns R_{ij}. For an input T, if $T = T^*$, where T^* is the challenge time, then $O_{TS\text{-}Release}$ returns $\perp$. Otherwise, $O_{TS\text{-}Release}$ returns a trapdoor s_T. For an input (upk, C, T), O_{DEC} returns $\perp$ if either (1) upk was not produced by the KeyGen algorithm executed by the challenger, or (2) $(upk, C, T) = (upk^*, C^*, T^*)$. Otherwise, O_{DEC} returns a decryption result M.

Definition 9 (IND-wCTCA Security at Second-level Ciphertext)**.** *We say that a (single-hop) TR-PRE scheme is IND-wCTCA-secure at second-level ciphertext if the advantage $Adv^{IND\text{-}wCTCA\text{-}2nd}_{\Pi,\mathcal{A}}(1^k)$ is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
&Adv^{IND\text{-}wCTCA\text{-}2nd}_{\Pi,\mathcal{A}}(1^k) = \\
&\quad \Big| \Pr\big[(params, ts_{priv}) \leftarrow \mathsf{Setup}(1^k); \\
&\quad Set\ O := \{O_{KeyGen}, O_{RE\text{-}ENC}, O_{RKGen}, O_{DEC}, \\
&\qquad\qquad O_{TS\text{-}Release}\}; \\
&\quad (M_0^*, M_1^*, T^*, upk^*, State) \leftarrow \mathcal{A}^{O}(params); \\
&\quad \mu \xleftarrow{\$} \{0,1\}; C^* \leftarrow \mathsf{Encrypt2}(params, upk^*, M_\mu^*, T^*); \\
&\quad \mu' \leftarrow \mathcal{A}^{O}(C^*, State); \mu = \mu'\big] - 1/2 \Big|
\end{aligned}
$$

Next, we define the IND-wCTCA security at first-level ciphertext. Oracles used in the following experiment are the same as that of the second-level one.

Definition 10 (IND-wCTCA Security at First-level Ciphertext)**.** *We say that a (single-hop) TR-PRE scheme is IND-wCTCA-secure at first-level ciphertext if the advantage $Adv^{IND\text{-}wCTCA\text{-}1st}_{\Pi,\mathcal{A}}(1^k)$ is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
&Adv^{IND\text{-}wCTCA\text{-}1st}_{\Pi,\mathcal{A}}(1^k) = \\
&\quad \Big| \Pr\big[(params, ts_{priv}) \leftarrow \mathsf{Setup}(1^k); \\
&\quad Set\ O := \{O_{KeyGen}, O_{RE\text{-}ENC}, O_{RKGen}, O_{DEC}, \\
&\qquad\qquad O_{TS\text{-}Release}\}; \\
&\quad (M_0^*, M_1^*, T^*, upk^*, State) \leftarrow \mathcal{A}^{O}(params); \\
&\quad \mu \xleftarrow{\$} \{0,1\}; C^* \leftarrow \mathsf{Encrypt1}(params, upk^*, M_\mu^*, T^*); \\
&\quad \mu' \leftarrow \mathcal{A}^{O}(C^*, State); \mu = \mu'\big] - 1/2 \Big|
\end{aligned}
$$

4. Proposed Scheme

In this section, we propose our TR-PRE scheme[†]. Our TR-PRE is based on the Libert-Vergnaud PRE [32], and the (IND-ID-CCA-secure [††]) Gentry IBE [26].

First, we explain how difficult is to construct TR-PRE (without random oracles) even if generic constructions of TRE [15]–[17], [34] are given. In Nakai et al.'s construction [34][†††] (based on IBE, Public Key Encryption (PKE), and sUF one-time signature), a ciphertext is represented as $(K_v, T, c_1, c_2, \sigma)$, where K_v is a signature verification key (paired with a signing key K_s), T is a release time, $c_1 = \mathsf{PKE.Enc}(upk, (K_v \| r))$, r is a random number chosen from the message space, upk is a user's public key, $c_2 = \mathsf{IBE.Enc}(T, (K_v \| (M \oplus r)))$, and $\sigma = \mathsf{Sign}(K_s, (T \| c_1 \| c_2))$. In this construction, T is regarded as the "identity" of the IBE scheme. Therefore, someone may think that it is not hard to construct TR-PRE by applying such generic constrictions of TRE, e.g., by replacing the PKE part into PRE and so on. However, when simply exchanging the underlying PKE scheme for a PRE scheme, σ cannot work after the proxy translates c_1 into c_1' (which can be decrypted by another user), since a "signed-message" c_1 has already been changed. Other generic constructions [15], [16] require random oracles, since these constructions apply the Fujisaki-Okamoto transformation [25]. In [17], a generic construction of TRE based on Security-Mediated Certificateless Encryption (SMCLE) was proposed. However, SMCLE is not a primitive tool (such as PKE, IBE, digital signatures, hash functions, and so on), and therefore "TRE combines PRE" is similar to "SMCLE combines PRE". From the above considerations, we need another structure to combine TRE and PRE schemes without random oracles.

The overview of our construction is as follows: As in the Nakai et al. construction, a release-time T is regarded as an identity of the underlying IBE scheme, and thus s_T is the private key of the Gently IBE. In addition, the part of ciphertexts of IBE and PRE containing a plaintext M are connected. More precisely, in the following construction, (C_3, C_5, C_6, C_7) is a ciphertext for a message M' of the Gentry IBE scheme, where $M' := M \cdot e(g,g)^{r_1}$, and (C_1, C_2, C_3, C_4) is a (part of) ciphertext for a message M'' of the Libert-Vergnaud PRE scheme, where $M'' := M \cdot e(g, h_1)^{r_2}$ (i.e., C_3 is commonly used from both IBE and PRE section). $e(g,g)^{r_1}$ is computed from a PRE section, and $e(g, h_1)^{r_2}$ is computed from an IBE section. Together with these elements, the cancel element $e(g,g)^{r_1} \cdot e(g, h_1)^{r_2}$ can be computed. In addition, a signature part of our construction is different from that of the Libert-Vergnaud PRE scheme. In the Libert-Vergnaud PRE scheme, a (one-time) signature is computed as $\sigma \leftarrow \mathsf{Sign}(K_s, (C_3, C_4))$. On the other hand, we include IBE ciphertexts (C_3, C_5, C_6, C_7) (and T also) in the signed message to bind all ciphertexts. This signed message does not change through the re-encryption procedure.

[†]Note that, we do not consider encrypting with distinct release times as in Cathalo et al. [13], since colluding receivers could decrypt the message without having the appropriate trapdoor.

[††]The CCA-secure Gentry IBE scheme also provides recipient anonymity. In the TR-PRE context, recipient anonymity property is not required. For the sake of clarity, we introduce the definition of IND-ID-CCA game and the Gentry IBE scheme in the Appendix.

[†††]Although this construction also handles pre-open capability, we omit the explanation of this property, since pre-open capability property is out-of-scope in our context.

So, by modifying the signed message above, σ works even after the proxy translates the second-level ciphertext.

Protocol 1. ***The proposed TR-PRE scheme***

$\mathsf{Setup}(1^k)$ *: Let* $(\mathbb{G}, \mathbb{G}_T)$ *be a bilinear group with prime order* p*,* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *be a bilinear map, and* $g, u, v, h_1, h_2, h_3 \in \mathbb{G}$ *be generators. Set the message space as* $\mathbb{G}_T$ *and the release time space as* $\mathbb{Z}_p$*. Select* $s \xleftarrow{\$} \mathbb{Z}_p^*$*, compute* $TS_{pub} = g^s$*, and output* $ts_{priv} = s$ *and* $params = (g, u, v, h_1, h_2, h_3, TS_{pub}, e(g,g), e(g,h_1), e(g,h_2), e(g,h_3), H)$*, where* $H : \{0,1\}^* \to \mathbb{Z}_p$ *is a cryptographic hash function chosen from a family of universal one-way hash functions (UOWHF) [35]*[†]*.*

$\mathsf{KeyGen}(params)$ *: For a user* U_i*, choose* $x_i \xleftarrow{\$} \mathbb{Z}_p$*, compute* $X_i = g^{x_i}$*, and output* $(upk_i, usk_i) = (X_i, x_i)$*.*

$\mathsf{TS\text{-}Release}(params, ts_{\mathsf{priv}}, T)$ *: For a release time* $T \in \mathbb{Z}_p$*, choose* $r_{T,1}, r_{T,2}, r_{T,3} \xleftarrow{\$} \mathbb{Z}_p^*$*, compute* $s_T = ((r_{T,1}, (h_1 \cdot g^{-r_{T,1}})^{\frac{1}{s-T}}), (r_{T,2}, (h_2 \cdot g^{-r_{T,2}})^{\frac{1}{s-T}}), (r_{T,3}, (h_3 \cdot g^{-r_{T,3}})^{\frac{1}{s-T}}))$*, and then output* s_T*.*

$\mathsf{Encrypt2}(params, upk_i, M, T)$ *: Let* $upk_i = X_i$*. For* $M \in \mathbb{G}_T$ *and* $T \in \mathbb{Z}_p$*, choose* $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ *and a one-time signature key pair* $(K_s, K_v) \leftarrow \mathsf{Sig.KeyGen}(1^k)$*, set* $C_1 := K_v$*, compute* $C_2 = X_i^{r_1}$*,* $C_3 = M \cdot e(g,g)^{r_1} \cdot e(g,h_1)^{r_2}$*,* $C_4 = (u^{K_v} \cdot v)^{r_1}$*,* $C_5 = (g^{-T} \cdot TS_{pub})^{r_2}$*,* $C_6 = e(g,g)^{r_2}$*, and* $C_7 = (e(g,h_2) \cdot e(g,h_3)^{\beta})^{r_2}$*, for* $\beta = H(C_3, C_5, C_6)$*. Then compute* $\sigma = \mathsf{Sign}(K_s, (C_3, C_4, C_5, C_6, C_7, T))$*. Output a second-level ciphertext* $C = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, \sigma, T)$*.*

$\mathsf{Encrypt1}(params, upk_i, M, T)$ *: Let* $upk_i = X_i$*. For* $M \in \mathbb{G}_T$ *and* $T \in \mathbb{Z}_p$*, choose* $r_1, r_2, t \xleftarrow{\$} \mathbb{Z}_p$ *and a one-time signature key pair* $(K_s, K_v) \leftarrow \mathsf{Sig.KeyGen}(1^k)$*, set* $C_1 := K_v$*, and compute* $C_2' = X_i^t$*,* $C_2'' = g^{1/t}$*,* $C_2''' = X_i^{r_1 t}$*,* $C_3 = M \cdot e(g,g)^{r_1} \cdot e(g,h_1)^{r_2}$*,* $C_4 = (u^{K_v} \cdot v)^{r_1}$*,* $C_5 = (g^{-T} \cdot TS_{pub})^{r_2}$*,* $C_6 = e(g,g)^{r_2}$*,* $C_7 = (e(g,h_2) \cdot e(g,h_3)^{\beta})^{r_2}$*, where* $\beta = H_2(C_3, C_5, C_6)$*, and* $\sigma = \mathsf{Sign}(K_s, (C_3, C_4, C_5, C_6, C_7, T))$*. Output a first-level ciphertext* $C = (C_1, C_2', C_2'', C_2''', C_3, C_4, C_5, C_6, C_7, \sigma, T)$*.*

$\mathsf{RKGen}(params, usk_i, upk_j)$ *: Let* $usk_i = x_i$ *and* $upk_j = X_j$*. Compute* $R_{ij} = X_j^{\frac{1}{x_i}} = g^{\frac{x_j}{x_i}}$*. Then output* R_{ij}*.*

$\mathsf{Re\text{-}Encrypt}(params, R_{ij}, upk_i, C)$ *: Let* $upk_i = X_i$*. For the second-level ciphertext* $C = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, \sigma, T)$*, check whether* C *was encrypted by using* X_i *or not by verifying the following:*

$$e(C_2, u^{C_1} \cdot v) \stackrel{?}{=} e(X_i, C_4), \text{ and}$$
$$\mathsf{Verify}(C_1, \sigma, (C_3, C_4, C_5, C_6, C_7, T)) \stackrel{?}{=} 1$$

If well-formed, the first-level ciphertext C' *is computed as follows: Choose* $t \xleftarrow{\$} \mathbb{Z}_p$*, compute* $C_2' = X_i^t$*,* $C_2'' = R_{ij}^{\frac{1}{t}}$*, and* $C_2''' = C_2^t$*, and output the first-level ciphertext* $C' = (C_1, C_2', C_2'', C_2''', C_3, C_4, C_5, C_6, C_7, \sigma, T)$*.*

$\mathsf{Decrypt}(params, usk, C, s_T)$ *:*

In the case of first-level ciphertexts *: Let* $(C_1, C_2', C_2'', C_2''', C_3, C_4, C_5, C_6, C_7, \sigma, T)$ *be the first-level ciphertext, and* $s_T = ((r_{T,1}, h_{T,1}), (r_{T,2}, h_{T,2}), (r_{T,3}, h_{T,3}))$ *be the trapdoor of* T*. Compute* $\beta = H(C_3, C_5, C_6)$ *and check*

$$e(C_2', C_2'') \stackrel{?}{=} e(X_j, g),$$
$$e(C_2''', u^{C_1} \cdot v) \stackrel{?}{=} e(C_2', C_4),$$
$$e(C_5, h_{T,2} h_{T,3}^{\beta}) \cdot C_6^{r_{T,2} + r_{T,3}\beta} \stackrel{?}{=} C_7, \text{ and}$$
$$\mathsf{Verify}(C_1, \sigma, (C_3, C_4, C_5, C_6, C_7, T)) \stackrel{?}{=} 1$$

If well-formed, compute

$$\begin{aligned} e(C_2'', C_2''')^{\frac{1}{x_j}} &= e(g^{\frac{x_j}{tx_i}}, g^{tx_i r_1})^{\frac{1}{x_j}} \\ &= e(g,g)^{r_1}, \\ e(C_5, h_{T,1}) \cdot C_6^{r_{T,1}} &= e((g^{-T} \cdot TS_{pub})^{r_2}, \\ (h_1 \cdot g^{-r_{T,1}})^{\frac{1}{s-T}}) \cdot e(g,g)^{r_{T,1} r_2} &= e(g,h_1)^{r_2}, \end{aligned}$$

and $C_3 / \{e(g,g)^{r_1} \cdot e(g,h_1)^{r_2}\} = M$*, and output* M*.*

In the case of second-level ciphertext *: Let* $(C_1, C_2, C_3, C_4, C_5, C_6, C_7, \sigma, T)$ *be a second-level ciphertext, and* $s_T = ((r_{T,1}, h_{T,1}), (r_{T,2}, h_{T,2}), (r_{T,3}, h_{T,3}))$ *be the trapdoor of* T*. Compute* $\beta = H(C_3, C_5, C_6)$ *and check*

$$e(C_2, u^{C_1} \cdot v) \stackrel{?}{=} e(X_i, C_4),$$
$$e(C_5, h_{T,2} h_{T,3}^{\beta}) \cdot C_6^{r_{T,2} + r_{T,3}\beta} \stackrel{?}{=} C_7, \text{ and}$$
$$\mathsf{Verify}(C_1, \sigma, (C_3, C_4, C_5, C_6, C_7, T)) \stackrel{?}{=} 1$$

If well-formed, compute

$$\begin{aligned} e(C_2, g)^{\frac{1}{x_i}} &= e(X_i^{r_1}, g)^{\frac{1}{x_i}} \\ &= e(g,g)^{r_1}, \\ e(C_5, h_{T,1}) \cdot C_6^{r_{T,1}} &= e((g^{-T} \cdot TS_{pub})^{r_2}, \\ &= e(g,h_1)^{r_2}, \text{ and} \end{aligned}$$

$C_3 / \{e(g,g)^{r_1} \cdot e(g,h_1)^{r_2}\} = M$*, and output* M*.*

5. Features of Our TR-PRE Scheme

5.1 Security Analysis

Here, we give proofs of our TR-PRE scheme.

Theorem 1. *Our TR-PRE scheme is IND-RCCA-secure at*

[†]Bellare and Rogaway [2] rename UOWHF to target collision resistant (TCR) hash functions. However, in this paper we use the name UOWHF according to the Gently IBE.

second-level ciphertext if the modified 3-QDBDH assumption holds, and the underlying one-time signature scheme is sUF.

Proof. This proof is similar to that of the Libert-Vergnaud PRE scheme. However, we cannot directly use the challenger of the Libert-Vergnaud PRE scheme in a black-box manner, since the signature part of our scheme is different from that of the Libert-Vergnaud PRE scheme. Therefore, we have to write down the detailed proof: Let $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{a^2}, B = g^b, Z)$ be a modified 3-QDBDH instance. We construct an algorithm $\mathcal{B}_1$ that can decide whether $Z = e(g,g)^{b/a^2}$ or not, by using an adversary $\mathcal{A}$ to break the IND-RCCA security at second-level ciphertext of our TR-PRE scheme.

Before constructing $\mathcal{B}_1$, we explain two cases in which we can break the sUF of the underlying one-time signature scheme: Let $C^* = (C_1^* = K_v^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, \sigma^*, T^*)$ be the challenge ciphertext. Let event_1 be the event that $\mathcal{A}$ issues a decryption query $(K_v^*, C_2', C_2'', C_2''', C_3, C_4, C_5, C_6, C_7, \sigma, T)$, where $\mathsf{Verify}(K_v^*, \sigma, (C_3, C_4, C_5, C_6, C_7, T)) = 1$. Let event_2 be the event that $\mathcal{A}$ issues a re-encryption query $(K_v^*, C_2, C_3, C_4, C_5, C_6, C_7, \sigma, T)$, where $\mathsf{Verify}(K_v^*, \sigma, (C_3, C_4, C_5, C_6, C_7, T)) = 1$. If either event_1 or event_2 occurs, then we can construct an algorithm (sey $\mathcal{B}_2$) that breaks sUF of the underlying one-time signature scheme.

From now, we construct an algorithm $\mathcal{B}_1$ that outputs a random bit and aborts when either event_1 or event_2 occurs. $\mathcal{B}_1$ computes $(K_s^*, K_v^*) \leftarrow \mathsf{Sig.KeyGen}(1^k)$, chooses $\alpha_1, \alpha_2 \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, and computes $u := A_1^{\alpha_1} = g^{a\alpha_1}$ and $v := A_1^{-\alpha_1 \cdot K_v^*} \cdot A_2^{\alpha_2} = g^{-a\alpha_1 K_v^* + a^2\alpha_2}$ (note that $u^{K_v} \cdot v = A_1^{\alpha_1(K_v - K_v^*)} \cdot A_2^{\alpha_2}$ will appear in a part of a ciphertext). $\mathcal{B}_1$ chooses $s \overset{\$}{\leftarrow} \mathbb{Z}_p$ as ts_{priv}, and $h_1, h_2, h_3 \overset{\$}{\leftarrow} \mathbb{Z}_p$, and computes $TS_{\mathrm{pub}} = g^s$.

Public/Secret Key Generation: For the target user, $\mathcal{B}_1$ chooses $x^* \overset{\$}{\leftarrow} \mathbb{Z}_p$, and computes $upk^* = X^* := A_2^{x^*}$. For an honest user U_h, $\mathcal{B}_1$ chooses $x_h \overset{\$}{\leftarrow} \mathbb{Z}_p$, and computes $upk_h = X_h := A_1^{x_h}$. For a corrupted user U_c, $\mathcal{B}_1$ chooses $x_c \overset{\$}{\leftarrow} \mathbb{Z}_p$ as usk_c, and computes $upk_c = X_c := g^{x_c}$.

Re-encryption Key Generation: For R_{c*}, $\mathcal{B}_1$ can compute $R_{c*} = (X^*)^{1/x_c}$, since $\mathcal{B}_1$ knows $usk_c = x_c$. For R_{h*}, $\mathcal{B}_1$ can compute $R_{h*} = A_1^{x^*/x_h} = g^{x^* a^2/(x_h a)}$. For R_{*h}, $\mathcal{B}_1$ can compute $R_{*h} = A_{-1}^{x_h/x^*} = g^{x_h a/(x^* a^2)}$. Note that R_{h*} and R_{*h} are valid re-encryption keys, since $usk^* = x^* a^2$ and $usk_h = x_h a$. For R_{hc}, $\mathcal{B}_1$ can compute $R_{hc} = A_{-1}^{x_c/x_h} = g^{x_c/(x_h a)}$. For R_{ch}, $\mathcal{B}_1$ can compute $R_{ch} = A_1^{x_h/x_c} = g^{x_h a/x_c}$. For $R_{cc'}$, $\mathcal{B}_1$ can compute $R_{cc'} = g^{x_c/x_{c'}}$, since $\mathcal{B}_1$ knows $usk_c = x_c$ and $usk_{c'} = x_{c'}$. For $R_{hh'}$, $\mathcal{B}_1$ can compute $R_{hh'} = g^{x_{h'}/x_h} = g^{x_{h'} a/(x_h a)}$.

From the above considerations, $\mathcal{B}_1$ can send $params = (g, u, v, h_1, h_2, h_3, TS_{\mathrm{pub}}, e(g,g), e(g,h_1), e(g,h_2), e(g,h_3), H)$, $Keys$, $ReKeys$, and ts_{priv} to $\mathcal{A}$, where $Keys := \{upk^*, \{upk_h\}, \{(upk_c, usk_c)\}\}$ and $ReKeys := \{\{R_{c*}\}, \{R_{h*}\}, \{R_{*h}\}, \{R_{hc}\}, \{R_{ch}\}, \{R_{cc'}\}, \{R_{hh'}\}\}$.

- When $\mathcal{A}$ issues $O_{RE\text{-}ENC}$ with an input (upk_i, upk_j, C), where $C = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, \sigma, T)$ is a second-level ciphertext, then if either upk_i or upk_j were not generated by $\mathcal{B}_1$, $\mathcal{B}_1$ returns $\perp$. If C is ill-formed, then $\mathcal{B}_1$ returns $\perp$. We consider the following three cases as follows:

  i is the target user and j is an honest user : $\mathcal{B}_1$ simply re-encrypts C by using R_{*h}.

  i is not the target user and j is an honest user : $\mathcal{B}_1$ simply re-encrypts C by using $R_{hh'}$ or R_{ch}.

  i is the target user and j is a corrupted user : If $C_1 = K_v^*$ (event_2), then $\mathcal{B}_1$ outputs a random bit, and aborts. Otherwise, $\mathcal{B}_1$ computes $C_2^{1/x^*} = ((X^*)^{r_1})^{1/x^*} = A_2^{r_1}$. Now $C_4 = (u^{K_v} \cdot v)^{r_1} = (A_1^{\alpha_1(K_v - K_v^*)} \cdot A_2^{\alpha_2})^{r_1}$. Therefore, $A_1^{r_1} = g^{ar_1} = \left(C_4 / (C_2^{1/x^*})^{\alpha_2}\right)^{1/(\alpha_1(K_v - K_v^*))}$ holds. $\mathcal{B}_1$ chooses $t, r_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$, sets $\tilde{t} := at/x_c$, and computes $C_2' = A_1^t = g^{at} = g^{x_c \cdot at/x_c} = g^{x_c \tilde{t}} = X_c^{\tilde{t}}$, $C_2'' = A_{-1}^{x_c/t} = g^{x_c/at} = g^{1/\tilde{t}}$, and $C_2''' = \{(C_4/(C_2^{1/x^*})^{\alpha_2})^{1/(\alpha_1(K_v - K_v^*))}\}^t = g^{ar_1 t} = g^{r_1 x_c \tilde{t}} = (X_c^{r_1})^t$.

- When $\mathcal{A}$ issues O_{DEC} with an input (upk_j, C, T), where C is the first-level ciphertext under upk_j, then if C is ill-formed, $\mathcal{B}_1$ returns $\perp$. In addition, if $C_1 = K_v^*$ and $(C_3, C_4, \sigma) = (C_3^*, C_4^*, \sigma^*)$ (this may occur after the challenge phase), then $\mathcal{B}_1$ returns $\perp$ since C us a derivative of (upk^*, C^*). If $upk_j = upk_c$, then $\mathcal{B}_1$ can decrypt C, since $\mathcal{B}_1$ knows usk_c. We consider the remaining two cases as follows:

  j is an honest user: Since $X_j = g^{ax_j}$, $e(C_2'', C_2''') = e(X_j, g)^{r_1} = e(g,g)^{ar_1 x_j}$ hold. In addition, $C_4 = (u^{K_v} \cdot v)^{r_1} = (A_1^{\alpha_1(K_v - K_v^*)} \cdot A_2^{\alpha_2})^{r_1} = g^{a\alpha_1 r_1 (K_v - K_v^*)} \cdot g^{a^2\alpha_2 r_1}$ holds. Therefore $\left(\frac{e(C_4, A_{-1})}{e(C_2'', C_2''')^{\alpha_2/x_j}}\right)^{\frac{1}{\alpha_1(K_v - K_v^*)}} = e(g,g)^{r_1}$ holds. By using x_j, $\mathcal{B}_1$ can compute $e(g,g)^{r_1}$. In addition, $\mathcal{B}_1$ can compute s_T, and $e(g,h_1)^{r_2}$ from (C_5, C_6, C_7). $\mathcal{B}_1$ returns $M = C_3 / \{e(g,g)^{r_1} \cdot e(g,h_1)^{r_2}\}$ to $\mathcal{A}$.

  j is the target user: If $C_1 = K_v^*$ (event_1), then $\mathcal{B}_1$ outputs a random bit, and aborts. Now $X_j = g^{x^* a^2}$. Therefore, $e(C_2'', C_2''') = (e, X_j, g)^{r_1} = e(g,g)^{a^2 r_1 x^*}$ hold. Since $C_4 = g^{a\alpha_1 r_1 (K_v - K_v^*)} \cdot g^{a^2\alpha_2 r_1}$ holds, $e(C_4, g) = e(g,g)^{a\alpha_1 r_1 (K_v - K_v^*)} \cdot e(g,g)^{a^2\alpha_2 r_1}$ holds. Therefore $\left(\frac{e(C_4,g)}{e(C'', C_2''')^{\alpha_2/x_j}}\right)^{\frac{1}{\alpha_1(K_v - K_v^*)}} = e(g,g)^{ar_1}$ holds. In addition to this, $e(C_4, A_{-1}) = e(g,g)^{\alpha_1 r_1 (K_v - K_v^*)} \cdot e(g,g)^{a\alpha_2 r_1}$ holds. $\mathcal{B}_1$ computes

$$\left(\frac{e(g,g)^{\alpha_1 r_1 (K_v - K_v^*)} \cdot e(g,g)^{a\alpha_2 r_1}}{(e(g,g)^{ar_1})^{\alpha_2}}\right)^{\frac{1}{\alpha_1(K_v - K_v^*)}} = e(g,g)^{r_1}$$

  In addition, $\mathcal{B}_1$ can compute s_T, and $e(g,h_1)^{r_2}$

$$
\begin{aligned}
&2Adv^{\text{modified 3-}QDBDH}_{\mathbb{G},\mathcal{B}_1}(1^k)\\
&= 2|\Pr[\mathcal{B}_1 \to 0 \wedge Z = e(g,g)^{b/a^2}] - \Pr[\mathcal{B}_1 \to 0 \wedge Z = e(g,g)^z]|\\
&= 2|\Pr[\mathcal{B}_1 \to 0|Z = e(g,g)^{b/a^2}]\Pr[Z = e(g,g)^{b/a^2}] - \Pr[\mathcal{B}_1 \to 0|Z = e(g,g)^z]\Pr[Z = e(g,g)^z]|\\
&= |\Pr[\mathcal{B}_1 \to 0|Z = e(g,g)^{b/a^2}] - \Pr[\mathcal{B}_1 \to 0|Z = e(g,g)^z]|\\
&= |1 - \Pr[\mathcal{B}_1 \to 1|Z = e(g,g)^{b/a^2}] - \Pr[\mathcal{B}_1 \to 0|Z = e(g,g)^z]|\\
&\geq |1 - (\tfrac{1}{2} + \mathcal{A}^{\text{IND-RCCA-2nd}}_{\Pi,\mathcal{A}}(1^k) - \Pr[\mathsf{forge}|Z = e(g,g)^{b/a^2}]) - (\tfrac{1}{2} - \Pr[\mathsf{forge}|Z = e(g,g)^z])|\\
&\geq (\mathcal{A}^{\text{IND-RCCA-2nd}}_{\Pi,\mathcal{A}}(1^k) - (\Pr[\mathsf{forge}|Z = e(g,g)^{b/a^2}] + \Pr[\mathsf{forge}|Z = e(g,g)^z]))\\
&= \mathcal{A}^{\text{IND-RCCA-2nd}}_{\Pi,\mathcal{A}}(1^k) - \Pr[\mathsf{forge}](\Pr[Z = e(g,g)^{b/a^2}] + \Pr[Z = e(g,g)^z])\\
&= \mathcal{A}^{\text{IND-RCCA-2nd}}_{\Pi,\mathcal{A}}(1^k) - \Pr[\mathsf{forge}]\\
&\geq \mathcal{A}^{\text{IND-RCCA-2nd}}_{\Pi,\mathcal{A}}(1^k) - Adv^{\text{one-time sUF-CMA}}_{\mathcal{B}_2}(1^k)
\end{aligned}
$$

Fig. 3 The probability estimations.

from (C_5, C_6, C_7). $\mathcal{B}_1$ returns $M = C_3/\{e(g,g)^{r_1} \cdot e(g,h_1)^{r_2}\}$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ sends (M_0^*, M_1^*, T^*) to $\mathcal{B}_1$. $\mathcal{B}_1$ chooses $r_2^* \xleftarrow{\$} \mathbb{Z}_p$, sets $C_1^* = K_v^*$, and computes $C_2^* = B^{x^*}$, $C_3 = M_\mu^* \cdot Z \cdot e(g,h_1)^{r_2^*}$, $C_4^* = B^{\alpha_2}$, $C_5^* = (g^{-T^*} \cdot TS_{\text{pub}})^{r_2^*}$, $C_6 = e(g,g)^{r_2^*}$, and $C_7 = e(g,h_2)^{r_2^*} \cdot e(g,h_3)^{r_2^*\beta}$, where $\beta = H(C_3^*, C_5^*, C_6^*)$, and $\sigma^* = \mathsf{Sign}(K_s^*, (C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, T^*))$.

Finally, $\mathcal{A}$ outputs μ'. $\mathcal{B}_1$ decides $Z = e(g,g)^{b/a^2}$ (i.e., $\mathcal{B}_1$ outputs 1) when $\mu' = \mu$, and Z is a random value (i.e., $\mathcal{B}_1$ outputs 0), otherwise. When $Z = e(g,g)^{b/a^2}$, $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, \sigma^*, T^*)$ is a valid ciphertext of M_μ^* with $r_1^* := b/a^2$. So, $\mathcal{A}$ has the advantage, and therefore

$$
\begin{aligned}
&\Pr[\mathcal{B}_1 \to 1|Z = e(g,g)^{b/a^2}]\\
&\geq \frac{1}{2} + \mathcal{A}^{\text{IND-RCCA-2nd}}(1^k) - \Pr[\mathsf{forge}|Z = e(g,g)^{b/a^2}]
\end{aligned}
$$

holds. Otherwise, if Z is a random value, M_μ^* is perfectly hidden by Z. So, $\mathcal{A}$ has no advantage, and therefore

$$
\Pr[\mathcal{B}_1 \to 0|Z = e(g,g)^z] \geq \frac{1}{2} - \Pr[\mathsf{forge}|Z = e(g,g)^z]
$$

holds. Finally, we estimate the advantage of $\mathcal{B}_1$. Let forge be the event that $\mathcal{B}_2$ breaks sUF-CMA of the underlying signature. From our simulation, $\Pr[\mathsf{forge}] = \Pr[\mathsf{event}_1 \vee \mathsf{event}_2] = Adv^{\text{one-time sUF-CMA}}_{\mathcal{B}_2}(1^k)$ hold. Then $Adv^{\text{IND-RCCA-2nd}}_{\Pi,\mathcal{A}}(1^k) \leq 2Adv^{\text{modified 3-}QDBDH}_{\mathbb{G},\mathcal{B}_1}(1^k) + Adv^{\text{one-time sUF-CMA}}_{\mathcal{B}_2}(1^k)$ holds from the estimations described in Fig. 3. □

Theorem 2. *Our TR-PRE scheme is IND-RCCA-secure at first-level ciphertext if the modified 3-QDBDH assumption holds, and the underlying one-time signature scheme is sUF.*

Proof. Let $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{a^2}, B = g^b, Z)$ be a modified 3-QDBDH instance. We construct an algorithm $\mathcal{B}_1$ that can decide whether $Z = e(g,g)^{b/a^2}$ or not, by using an adversary $\mathcal{A}$ to break the IND-RCCA security at first-level ciphertext of our TR-PRE scheme.

As in the second-level ciphertext case, we explain the case in which we can break the sUF of the underlying one-time signature scheme: Let $C^* = (C_1^* = K_v^*, C_2'^*, C_2''^*, C_2'''^*, C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, \sigma^*, T^*)$ be the challenge ciphertext. Let event be the event that $\mathcal{A}$ issues a decryption query $(K_v^*, C_2', C_2'', C_2''', C_3, C_4, C_5, C_6, C_7, \sigma, T)$, where $\mathsf{Verify}(K_v^*, \sigma, (C_3, C_4, C_5, C_6, C_7, T)) = 1$. If event occurs, then we can construct an algorithm that breaks sUF of the underlying one-time signature scheme.

From now, we construct an algorithm $\mathcal{B}_1$ that outputs a random bit and aborts when event occurs. $\mathcal{B}_1$ computes $(K_s^*, K_v^*) \leftarrow \mathsf{Sig.KeyGen}(1^k)$, chooses $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_p^*$, and computes $u := A_1^{\alpha_1} = g^{a\alpha_1}$ and $v := A_1^{-\alpha_1 \cdot K_v^*} \cdot A_2^{\alpha_2} = g^{-a\alpha_1 K_v^* + a^2\alpha_2}$ (note that $u^{K_v} \cdot v = A_1^{\alpha_1(K_v - K_v^*)} \cdot A_2^{\alpha_2}$ will appear in a part of a ciphertext). $\mathcal{B}_1$ chooses $s \xleftarrow{\$} \mathbb{Z}_p$ as ts_{priv}, and $h_1, h_2, h_3 \xleftarrow{\$} \mathbb{Z}_p$, and computes $TS_{\text{pub}} = g^s$.

Public/Secret Key Generation: For an honest user U_h, $\mathcal{B}_1$ chooses $x_h \xleftarrow{\$} \mathbb{Z}_p$, and computes $upk_h = X_h := g^{x_h}$. For a corrupted user U_c, $\mathcal{B}_1$ chooses $x_c \xleftarrow{\$} \mathbb{Z}_p$ as usk_c, and computes $upk_c = X_c := g^{x_c}$. For the target user, $\mathcal{B}_1$ sets $upk^* = X^* := A_1$.

Re-encryption Key Generation: $\mathcal{B}_1$ can compute all re-encryption keys as follows. For R_{hc}, $\mathcal{B}_1$ can compute $R_{hc} = g^{x_c/x_h}$. For R_{ch}, $\mathcal{B}_1$ can compute $R_{ch} = g^{x_h/x_c}$. For $R_{cc'}$, $\mathcal{B}_1$ can compute $R_{cc'} = g^{x_c/x_{c'}}$. For $R_{hh'}$, $\mathcal{B}_1$ can compute $R_{hh'} = g^{x_{h'}/x_h}$. For R_{c*}, $\mathcal{B}_1$ can compute $R_{c*} = A_1^{1/x_c} = g^{a/x_c}$. For R_{h*}, $\mathcal{B}_1$ can compute $R_{h*} = A_1^{1/x_h} = g^{a/x_h}$. For R_{*h}, $\mathcal{B}_1$ can compute $R_{*h} = A_{-1}^{x_h} = g^{x_h/a}$. For R_{*c}, $\mathcal{B}_1$ can compute $R_{*c} = A_{-1}^{x_c} = g^{x_c/a}$.

From the above considerations, $\mathcal{B}_1$ can send $params = (g, u, v, h_1, h_2, h_3, TS_{\text{pub}}, e(g,g), e(g,h_1), e(g,h_2), e(g,h_3), H)$, $Keys$, $ReKeys$, and ts_{priv} to $\mathcal{A}$, where $Keys := \{upk^*, \{upk_h\}, \{(upk_c, usk_c)\}\}$ and $ReKeys := \{\{R_{c*}\}, \{R_{h*}\}, \{R_{*h}\}, \{R_{*c}\}, \{R_{hc}\}, \{R_{ch}\}, \{R_{cc'}\}, \{R_{hh'}\}\}$.

When $\mathcal{A}$ issues $\mathcal{O}_{DEC}$ with an input (upk_j, C, T), where C is the first-level ciphertext under upk_j, then if C is ill-formed, $\mathcal{B}_1$ returns $\perp$. In the case that j is not the target user (i.e., $upk_j \neq upk^*$), then $\mathcal{B}_1$ can decrypt C using x_h

or x_c. So, we assume that $upk_j = upk^*$. If $e(C_2'', C_2''') = e(C_2''^*, C_2'''^*)$ (this may occur after the challenge phase), then $\mathcal{B}_1$ returns $\perp$, since (upk_j, C) is a derivative of (upk^*, C^*). Now for (unknown) exponents $r_1, t \in \mathbb{Z}_p^*$, $X_j = A_1$, $C_2' = A_1^t$, $C_2'' = g^{1/t}$, and $C_2''' = A_1^{r_1 t}$ hold. From $e(C_2'', C_2''') = e(X_j, g)^{r_1} = e(g,g)^{ar_1}$ and $C_4 = (u^{K_v} \cdot v)^{r_1} = (A_1^{\alpha_1(K_v - K_v^*)} \cdot A_2^{\alpha_2})^{r_1} = g^{a\alpha_1 r_1 (K_v - K_v^*)} \cdot g^{a^2 \alpha_2 r_1}$, $\left(\frac{e(C_4, A_{-1})}{e(C_2'', C_2''')^{\alpha_2}}\right)^{\frac{1}{\alpha_1(K_v - K_v^*)}} = e(g,g)^{r_1}$ holds. In addition, $\mathcal{B}_1$ can compute s_T, and $e(g, h_1)^{r_2}$ from (C_5, C_6, C_7). $\mathcal{B}_1$ returns $M = C_3 / \{e(g,g)^{r_1} \cdot e(g, h_1)^{r_2}\}$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ sends (M_0^*, M_1^*, T^*) to $\mathcal{B}_1$. $\mathcal{B}_1$ chooses $t^*, r_2^* \xleftarrow{\$} \mathbb{Z}_p$, sets $C_1^* = K_v^*$, and computes $C_2'^* = A_2^{t^*}$, $C_2''^* = A_{-1}^{1/t^*}$, $C_2'''^* = B^{t^*}$, $C_3 = M_\mu^* \cdot Z \cdot e(g, h_1)^{r_2^*}$, $C_4^* = B^{\alpha_2}$, $C_5^* = (g^{-T^*} \cdot TS_{\text{pub}})^{r_2^*}$, $C_6 = e(g,g)^{r_2^*}$, and $C_7 = e(g, h_2)^{r_2^*} \cdot e(g, h_3)^{r_2^* \beta}$, where $\beta = H(C_3^*, C_5^*, C_6^*)$, and $\sigma^* = \mathsf{Sign}(K_s^*, (C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, T^*))$. When $Z = e(g,g)^{b/a^2}$, $C^* = (C_1^*, C_2'^*, C_2''^*, C_2'''^*, C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, \sigma^*, T^*)$ is a valid ciphertext of M_μ^* with $r_1^* := b/a^2$. Otherwise, if Z is a random value, M_μ^* is perfectly hidden by Z. Therefore, $\mathcal{B}_1$ decides $Z = e(g,g)^{b/a^2}$ when $\mu' = \mu$, and Z is a random value, otherwise.

As in the case of the second-ciphertext one, $Adv_{\Pi,\mathcal{A}}^{\text{IND-RCCA-1st}}(1^k) \leq 2Adv_{\mathbb{G},\mathcal{B}_1}^{\text{modified 3-}QDBDH}(1^k) + Adv_{\mathcal{B}_2}^{\text{one-time sUF-CMA}}(1^k)$ holds. □

Theorem 3. *Our TR-PRE scheme is IND-wCTCA-secure at second/first-level ciphertext if the truncated decisional q-ABDHE assumption holds, H is chosen from a UOWHF family, and the underlying one-time signature scheme is sUF.*

Proof. The roadmap of the proof is described as follows. We can use the challenger of the IND-ID-CCA game of the Gentry IBE scheme C in a black-box manner. The simulator $\mathcal{B}_1$ chooses all PRE-related parameters (incl. all user's secret keys), and can use C when $O_{TS\text{-}Release}$ and O_{DEC} are issued by $\mathcal{A}$. Especially, $\mathcal{B}_1$ can decrypt (upk, C, T) if an element (canceled by the TRE section) $e(g, h_1)^{r_2}$ is computed by C, since $\mathcal{B}_1$ knows all user's secret keys. Since the Gentry IBE scheme is IND-ID-CCA secure if the truncated decisional q-ABDHE assumption holds and H is chosen from a UOWHF family†, the theorem holds. For the sake of clarity, we introduce the definition of IND-ID-CCA game and the Gentry IBE scheme in the Appendix.

As in the IND-RCCA case, we explain the case in which we can break the sUF of the underlying one-time signature scheme: Let event be the event that $\mathcal{A}$ issues a decryption query where $\mathsf{Verify}(K_v^*, \sigma, (C_3, C_4, C_5, C_6, C_7, T)) = 1$. If event occurs, then we can construct an algorithm (say $\mathcal{B}_2$) that breaks sUF of the underlying one-time signature scheme.

From now, we construct an algorithm $\mathcal{B}_1$ that outputs a random bit and aborts when event occurs. First, C sends $ibe.pk = (g, g_1, h_1, h_2, h_3, H)$ to $\mathcal{B}_1$. $\mathcal{B}_1$ computes $(K_s^*, K_v^*) \leftarrow \mathsf{Sig.KeyGen}(1^k)$, sets $TS_{pub} = g_1$, chooses $u, v \xleftarrow{\$} \mathbb{G}$, and sends $params = (g, u, v, h_1, h_2, h_3, TS_{\text{pub}}, e(g,g), e(g,h_1), e(g,h_2), e(g,h_3), H)$ to $\mathcal{A}$.

- For O_{KeyGen} issued by $\mathcal{A}$, $\mathcal{B}_1$ executes $(X, x) \leftarrow \mathsf{KeyGen}(params)$, and sends $(upk, usk) = (X, x)$ to $\mathcal{A}$.
- For $O_{RE\text{-}ENC}$ and O_{RKGen}, $\mathcal{B}_1$ can answer the query since $\mathcal{B}_1$ has all secret keys usk.
- When $\mathcal{A}$ issues $O_{TS\text{-}Release}$ with an input T, $\mathcal{B}_1$ forwards T to C as a $\mathcal{EXTRACT}$ query, obtains s_T, and sends s_T to $\mathcal{A}$.
- When $\mathcal{A}$ issues O_{DEC} with an input (upk, C, T), if upk was not generated by $\mathcal{B}_1$, $\mathcal{B}_1$ returns $\perp$. If C is ill-formed, $\mathcal{B}_1$ returns $\perp$ (note that $\mathcal{B}_1$ cannot check the equation $e(C_5, h_{T,2} h_{T,3}^{\beta}) \cdot C_6^{r_{T,2} + r_{T,3}\beta} \stackrel{?}{=} C_7$ if the corresponding s_T has not been appeared. However, since the validity check of the IBE section can turn over the decryption oracle $\mathcal{DEC}$, here $\mathcal{B}_1$ just check the validity of the remaining equations (the PRE section and the one-time signature). By using usk (paired with upk), $\mathcal{B}_1$ decrypts the PRE section, and obtains $e(g,g)^{r_1}$ for an unknown exponent $r_1 \in \mathbb{Z}_p^*$. In addition, $\mathcal{B}_1$ sends (C_3, C_5, C_6, C_7, T) to C as a $\mathcal{DEC}$ query, obtains M' from C, and sends $M'/e(g,g)^{r_1}$ to $\mathcal{A}$. Note that if C returns $\perp$ (i.e., (C_3, C_5, C_6, C_7, T) is not a valid IBE ciphertext), then $\mathcal{B}_1$ also returns $\perp$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ sends $(M_0^*, M_1^*, T^*, upk^* := X^*)$ to $\mathcal{B}_1$. Next, we explain the IND-wCTCA-1st case and the IND-wCTCA-2nd case, respectively.

The first-level ciphertext: $\mathcal{B}_1$ chooses $r_1^*, t^* \xleftarrow{\$} \mathbb{Z}_p$, set $C_1^* := K_v^*$, and compute $e(g,g)^{r_1^*}$, $C_2'^* = X^{*t^*}$, $C_2''^* = g^{1/t^*}$, $C_2'''^* = X^{*r_1^* t^*}$, and $C_4^* = (u^{K_v^*} \cdot v)^{r_1^*}$. $\mathcal{B}_1$ sets $(M_0', M_1') := (M_0^* \cdot e(g,g)^{r_1^*}, M_1^* \cdot e(g,g)^{r_1^*})$ as the challenge message of the Gentry IBE, and sends $((M_0')^*, (M_1')^*)$ to C. C gives the challenge ciphertext of the Gentry IBE $(C_{IBE,1}^*, C_{IBE,2}^*, C_{IBE,3}^*, C_{IBE,4}^*)$. $\mathcal{B}_1$ sets $C_3^* := C_{IBE,3}^*$, $C_5^* := C_{IBE,1}^*$, $C_6^* := C_{IBE,2}^*$, and $C_7^* := C_{IBE,4}^*$, and computes $\sigma^* \leftarrow \mathsf{Sign}(K_s^*, (C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, T^*))$, and sends $C^* = (C_1^*, C_2'^*, C_2''^*, C_2'''^*, C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, \sigma^*, T^*)$.

The second-level ciphertext: This is the same as the of the first-level ciphertext case, except $\mathcal{B}_1$ computes $C_2^* = X^{*r_1^*}$ instead of $(C_2'^*, C_2''^*, C_2'''^*)$.

Note that $\mathcal{B}_1$ cannot decrypt the challenge ciphertext C^*, since the TRE part of the C^* is the challenge ciphertext of the Gentry IBE scheme. Finally, $\mathcal{A}$ outputs μ'. $\mathcal{B}_1$ outputs μ' to C as the guessing bit. So, $Adv_{\Pi,\mathcal{A}}^{\text{IND-wCTCA-1st}}(1^k) \leq Adv_{\text{Gentry IBE},\mathcal{B}_1}^{\text{IND-ID-CCA}}(1^k) + Adv_{\mathcal{B}_2}^{\text{one-time sUF-CMA}}(1^k)$ and $Adv_{\Pi,\mathcal{A}}^{\text{IND-wCTCA-2nd}}(1^k) \leq Adv_{\text{Gentry IBE},\mathcal{B}_1}^{\text{IND-ID-CCA}}(1^k) + Adv_{\mathcal{B}_2}^{\text{one-time sUF-CMA}}(1^k)$ hold. □

†Although Gentry does not include the state of the hash function into the theorem of his IBE scheme, the universal onewayness of H is required in the proof of the Gentry IBE scheme. So, in this paper, we explicitly require that H is chosen from a UOWHF family.

Table 1 Efficiency comparison.

	Enc. Cost (single recipient)	Enc. Cost (N recipients)	Re-Enc. Cost	Dec. Cost (first)
Cathalo et al. TRE [13]	$ME(\mathbb{G}) + ME(\mathbb{G}_T)$ $+2BM$	$N(ME(\mathbb{G}) + 2BM)$ $+ME(\mathbb{G}_T)$	-	-
Our TR-PRE	$3ME(\mathbb{G})$ $+3ME(\mathbb{G}_T)$ + Sign	$3ME(\mathbb{G})$ $+3ME(\mathbb{G}_T)$ + Sign	$N(4ME(\mathbb{G})$ $+2BM$ + Sig.ver)	$2ME(\mathbb{G}) + 4ME(\mathbb{G}_T)$ $+6BM$ + Sig.ver

	Dec. Cost (second)	Ciphertext Size	Standard Model
Cathalo et al. TRE [13]	$ME(\mathbb{G})$ $+ME(\mathbb{G}_T) + BM$	$\|M\| + k + \|\mathbb{G}\|$	No
Our TR-PRE	$2ME(\mathbb{G}) + 4ME(\mathbb{G}_T)$ $+6BM$ + Sig.ver	$\|\sigma\| + \|K_v\| + 5\|\mathbb{G}\| + 3\|\mathbb{G}_T\|$	Yes

5.2 Efficiency Comparisons

Here, we compare our TR-PRE scheme and the TRE scheme proposed by Cathalo-Libert-Quisquater TRE [13] in Table 1. Note that, as mentioned before, in the Cathalo et al. TRE, if each ciphertext (for a user U_i) is represented as $(C_i, (M\|\text{random nonce}) \oplus K, T)$, then actual transferred ciphertext size is constant. So, we estimate the communication cost (i.e., the size of ciphertext per each receiver) of the Cathalo et al. TRE with such customized ciphertext form. Since other TRE schemes do not consider the multiple recipients case, we omit these schemes from Table 1.

$ME(\mathbb{G})$ and $ME(\mathbb{G}_T)$ denote the computational cost of multi-exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$, respectively. BM denotes that of one bilinear map computation. $|\mathbb{G}|$ and $\left|\mathbb{Z}_p\right|$ denotes the bit-length of the representation of a element of $\mathbb{G}$ and $\mathbb{Z}_p$, respectively. $|M|$ denotes the bit-length of the plaintext space, $|\sigma|$ denotes the bit-length of the signature, and $|K_v|$ denotes the bit-length of the verification key. Note that k (appeared in the Cathalo et al. TRE) is the security parameter which indicates the size of the random nonce. We omit the costs of both the re-encryption and decryption of the first level ciphertext from the Cathalo et al. TRE estimation. In addition, the ciphertext of the Cathalo et al. TRE is regarded as the second-level ciphertext, since it is not applied the proxy re-encryption procedure.

Due to random oracles, the Cathalo et al. TRE achieves highly efficient construction and much smaller ciphertext size compared with our TR-PRE scheme[†]. However, it is desirable to construct cryptographic schemes without random oracles even if efficient cryptographic schemes can be easily achieved in the random oracle model. For example, Canetti et al. [10] show that there exist signature and encryption schemes, which are secure in the random oracle model, but are insecure when random oracles are replaced with actual hash functions. Constructing protocols in standard model is thus important in real-life applications since there is no known hash function that is perfectly random. In addition, encryption costs of the Cathalo et al. TRE linearly depend on N (it can be a serious problem when N becomes large), whereas no costs depend on N from the encryptor's/decryptor's point of view in our TR-PRE scheme. This is a superior point of our TR-PRE scheme compared with the Cathalo et al. TRE scheme.

5.3 Is Technique of Attribute-Based Proxy Re-Encryption Applicable to TR-PRE?

Liang et al. proposed AB-PRE [31]. Considering a release-time T (and identity of user also) as an attribute, it is expected that TR-PRE is implied by AB-PRE. However, we show that AB-PRE is not suitable for constructing TRE scheme with fully constant costs as follows.

In AB-PRE, as in Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [4], a decryption key is assigned with a set of attributes, and a ciphertext is assigned with an access structure. The proxy translates a ciphertext C assigned with an access structure AS into the re-encrypted ciphertext assigned with another access structure AS'. Each user is given the corresponding decryption key assigned with certain attributes by a trusted key generation authority (KGA). Then, for example, by indicating $AS = (T \wedge U_1)$ as the access structure of the second-level ciphertext and $AS' = (T \wedge U_2)$ as the access structure of the first-level ciphertext, AB-PRE might work like TRE with fully constant costs from the encryptor's and decryptor's point of view. However, due to the functionality of AB-PRE, KGA can know all plaintext, and therefore KGA is modeled as fully trusted. As mentioned in Sect. 1, the condition that no authority can decrypt ciphertexts should be satisfied as in TRE. Thus, AB-PRE is not suitable for constructing TRE scheme with fully constant costs.

6. Applications of TR-PRE

By using TR-PRE, we can achieve a multicast secure communication with release time indication[††]. For example,

[†]Note that $2BM$ containing the encryption costs of the Cathalo et al. TRE is for checking whether the public key upk is valid or not, namely, for $upk = (X, Y)$, the verification $e(X, TS_{pub}) \stackrel{?}{=} e(g, Y)$ is required. Therefore, this verification is required for the first communication only.

[††]Actually, as applications of PRE schemes, e-mail systems based on PRE have been proposed, such as [6], [28]–[30]. By using TR-PRE as a building tool of these e-mail systems, we can achieve e-mail systems with release time indication.

in an on-line examination, an examiner sends encrypted e-mails to each examinee, and each examination can be opened at the same time. Compared with the case of applying TRE, we can reduce the encryption costs of the examiner. Compared with the case of applying public key encryption with recipient-to-recipient encryption, we can achieve fairer examination, since each examinee can decrypt the corresponding encrypted e-mail, simultaneously.

By applying the fact that a trapdoor s_T can be used commonly for plural ciphertexts assigned with T, we can apply TR-PRE to the case where huge encrypted data (e.g., digital movies) are transferred all over the world, and their release time (e.g., release date) is indicated. Even if a conventional PKE scheme is applied, encrypted contents (which are encrypted by each recipient's public key) cannot be delivered before the release date, since the contents might be leaked though release date has not been passed. If such encrypted contents are delivered right before the release date, then there is an possibility of delaying release time, since huge encrypted data need to be transferred. By using a conventional TRE scheme, we can achieve that encrypted contents (which is encrypted by each recipient's public key) can be delivered before the release date with reasonable margin, and s_T is delivered right before the release date. However, since there is no TRE scheme with fully constant costs, a distributor is subject to huge amount of computational costs. On the other hand, our TR-PRE achieves fully constant costs from the encryptor's/decryptor's point of view. So, we can construct an efficient fairly-opened multi-cast cryptosystem with release time indication by applying TR-PRE.

7. Conclusion

In this paper, to achieve TRE with fully constant costs from the encryptor's/decryptor's point of view, we propose a TR-PRE scheme based on the Libert-Vergnaud PRE [32] and the Gentry IBE [26]. An encryptor can foist N-dependent computation costs on the proxy, and therefore any factor (ciphertext size, computational complexity of encryption/decryption, and public/secret key size) does not depend on N, except the proxy re-encryption costs. TR-PRE works like TRE with fully constant costs from the encryptor's and decryptor's point of view. TR-PRE functionality can be applied to efficient multicast secure communication with a release time indication.

In cloud computing environments, users do not have to grasp the actual data storage of some services, and therefore data management becomes more and more difficult. Usually, *access control of data* and *encryption of data* are different technologies. Therefore, TRE (ABE [4] and searchable encryption [9] are also another examples) is suitable in cloud computing environments, since the access control function is included in the encrypted data itself. In PRE, access control (namely, who has decryption rights) may be complicated and hard to manage, when the number of users becomes large. TR-PRE is valuable in adding an access control function into encrypted (and re-encrypted) data itself. This feature is suitable for data management (e.g., when ciphertexts can be decrypted) in cloud computing environments.

References

[1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," ACM Trans. Inf. Syst. Secur., vol.9, no.1, pp.1–30, 2006.
[2] M. Bellare and P. Rogaway, "Collision-resistant hashing: Towards making UOWHFs practical," CRYPTO, pp.470–484, 1997.
[3] M. Bellare and S. Shoup, "Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles," Public Key Cryptography, pp.201–216, 2007.
[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," IEEE Symposium on Security and Privacy, pp.321–334, 2007.
[5] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," EUROCRYPT, pp.127–144, 1998.
[6] R. Bobba, J. Muggli, M. Pant, J. Basney, and H. Khurana, "Usable secure mailing lists with untrusted servers," IDtrust, pp.103–116, 2009.
[7] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," EUROCRYPT, pp.223–238, 2004.
[8] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," EUROCRYPT, pp.440–456, 2005.
[9] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," EUROCRYPT, pp.506–522, 2004.
[10] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," J. ACM, vol.51, no.4, pp.557–594, 2004.
[11] R. Canetti, S. Halevi, and J. Katz, "Chosen-ciphertext security from identity-based encryption," EUROCRYPT, pp.207–222, 2004.
[12] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," ACM Conference on Computer and Communications Security, pp.185–194, 2007.
[13] J. Cathalo, B. Libert, and J.-J. Quisquater, "Efficient and non-interactive timed-release encryption," ICICS, pp.291–303, 2005.
[14] K. Chalkias, D. Hristu-Varsakelis, and G. Stephanides, "Improved anonymous timed-release encryption," ESORICS, pp.311–326, 2007.
[15] J.H. Cheon, N. Hopper, Y. Kim, and I. Osipkov, "Timed-release and key-insulated public key encryption," Financial Cryptography, pp.191–205, 2006.
[16] J.H. Cheon, N. Hopper, Y. Kim, and I. Osipkov, "Provably secure timed-release public key encryption," ACM Trans. Inf. Syst. Secur., vol.11, no.2, 2008.
[17] S.S.M. Chow, V. Roth, and E.G. Rieffel, "General certificateless encryption and timed-release encryption," SCN, pp.126–143, 2008.
[18] S.S.M. Chow and S.-M. Yiu, "Timed-release encryption revisited," ProvSec, pp.38–51, 2008.
[19] C.-K. Chu and W.-G. Tzeng, "Identity-based proxy re-encryption without random oracles," ISC, pp.189–202, 2007.
[20] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," CRYPTO, pp.13–25, 1998.
[21] C. Delerablée, P. Paillier, and D. Pointcheval, "Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys," Pairing, pp.39–59, 2007.
[22] A.W. Dent and Q. Tang, "Revisiting the security model for timed-release encryption with pre-open capability," ISC, pp.158–174, 2007.
[23] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," Public Key Cryptography, pp.416–431,

2005.
[24] K. Emura, A. Miyaji, and K. Omote, "A timed-release proxy re-encryption scheme and its application to fairly-opened multicast communication," ProvSec, pp.200–213, 2010.
[25] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," CRYPTO, pp.537–554, 1999.
[26] C. Gentry, "Practical identity-based encryption without random oracles," EUROCRYPT, pp.445–464, 2006.
[27] M. Green and G. Ateniese, "Identity-based proxy re-encryption," ACNS, pp.288–306, 2007.
[28] H. Khurana and H.-S. Hahm, "Certified mailing lists," ASIACCS, pp.46–58, 2006.
[29] H. Khurana, J. Heo, and M. Pant, "From proxy encryption primitives to a deployable secure-mailing-list solution," ICICS, pp.260–281, 2006.
[30] H. Khurana, A.J. Slagell, and R. Bonilla, "SELS: A secure e-mail list service," SAC, pp.306–313, 2005.
[31] X. Liang, Z. Cao, H. Lin, and J. Shao, "Attribute based proxy re-encryption with delegating capabilities," ASIACCS, pp.276–286, 2009.
[32] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," Public Key Cryptography, pp.360–379, 2008.
[33] T.C. May, "Time-release crypto," Unpublished manuscript, 1993.
[34] Y. Nakai, T. Matsuda, W. Kitada, and K. Matsuura, "A generic construction of timed-release encryption with pre-open capability," IWSEC, pp.53–70, 2009.
[35] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," STOC, pp.33–43, 1989.

Appendix

In this Appendix, we introduce the security definition of IND-ID-CCA security and the Gentry IBE scheme for the sake of clarity of the proof of Theorem 3.

An IBE scheme Π consists of four algorithms, $\mathsf{IBE.Setup}$, $\mathsf{IBE.Extract}$, $\mathsf{IBE.Enc}$ and $\mathsf{IBE.Dec}$. The public key $ibe.pk$ and the master key $ibe.mk$ are given by executing $\mathsf{IBE.Setup}(1^k)$. For an identity $ID \in \mathcal{ID}$, where $\mathcal{ID}$ is the identity space (and $\mathcal{ID} = \mathbb{Z}_p$ in the Gentry IBE scheme), a secret key corresponding to ID s_{ID} is given by executing $\mathsf{IBE.Extract}(ibe.pk, ibe.mk, ID)$. For a message $M \in \mathcal{M}_{IBE}$ and $ID \in \mathcal{ID}$, where $\mathcal{M}_{IBE}$ is the message space of IBE, an encryptor runs $\mathsf{IBE.Enc}(ibe.pk, ID, M)$, and obtains a ciphertext C_{IBE}. The message M is computed by executing $\mathsf{IBE.Dec}(s_{ID}, C_{IBE})$.

Next, we define the security experiment of IBE under chosen ciphertext attack (IND-ID-CCA) as follows.

Definition 11 (IND-ID-CCA)**.** *An IBE scheme is said to be IND-ID-CCA secure if the advantage is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
&Adv^{IND\text{-}ID\text{-}CCA}_{\Pi,\mathcal{A}}(1^k)\\
&:= \Big|\Pr\big[(ibe.pk, ibe.mk) \leftarrow \mathsf{IBE.Setup}(1^k);\\
&\quad (M_0^*, M_1^*, ID^*, State) \leftarrow \mathcal{A}^{\mathcal{EXTRACT},\mathcal{DEC}}(ibe.pk);\\
&\quad \mu \overset{\$}{\leftarrow} \{0,1\};\ C^*_{IBE} \leftarrow \mathsf{IBE.Enc}(ibe.pk, ID^*, M^*_\mu);\\
&\quad \mu' \leftarrow \mathcal{A}^{\mathcal{EXTRACT},\mathcal{DEC}}(C^*_{IBE}, State);\ \mu = \mu'\big] - 1/2\Big|
\end{aligned}
$$

Let $\mathcal{EXTRACT}$ be an extract oracle, where, for input of an identity ID, it returns the corresponding secret key s_{ID}. Note that ID^* is not allowed to input to $\mathcal{EXTRACT}$. Let $\mathcal{DEC}$ be a decryption oracle, where, for input of a ciphertext C and an identity ID, it returns the corresponding plaintext M or $\perp$ according to the $\mathsf{IBE.Dec}$ *algorithm. Note that (ID^*, C^*_{IBE}) is not allowed to input to $\mathcal{DEC}$.*

Next, we introduce the Gentry IBE scheme as follows.

Protocol 2 (The IND-ID-CCA secure Gentry IBE)**.**

$\mathsf{IBE.Setup}(1^k)$**:** *Set the message space $\mathcal{M}_{IBE} = \mathbb{G}_T$ and the identity space $\mathcal{ID} = \mathbb{Z}_p$. Choose generators $g, h_1, h_2, h_3 \overset{\$}{\leftarrow} \mathbb{G}$, $s \overset{\$}{\leftarrow} \mathbb{Z}_p$, and a hash function $H : \{0,1\}^* \to \mathbb{Z}_p$ (chosen from a UOWHF family [35]), compute $g_1 = g^s$, and output $ibe.pk = (g, g_1, h_1, h_2, h_3, H)$ and $ibe.mk = s$.*

$\mathsf{IBE.Extract}(ibe.pk, ibe.mk, ID)$**:** *Choose $r_{ID,1}$, $r_{ID,2}$, $r_{ID,3} \overset{\$}{\leftarrow} \mathbb{Z}_p$, compute $s_{ID} = ((r_{ID,1}, h_{ID,1} = (h_1 g^{-r_{ID,1}})^{\frac{1}{(s-ID)}}, (r_{ID,2}, h_{ID,2} = (h_2 g^{-r_{ID,2}})^{\frac{1}{(s-ID)}}, (r_{ID,3}, h_{ID,3} = (h_3 g^{-r_{ID,3}})^{\frac{1}{(s-ID)}})$, and output s_{ID}.*

$\mathsf{IBE.Enc}(ibe.pk, ID, M)$**:** *For a plaintext $M \in \mathbb{G}$, choose $r \overset{\$}{\leftarrow} \mathbb{Z}_p$, and compute $C_{IBE,1} = (g_1 g^{-ID})^r$, $C_{IBE,2} = e(g,g)^r$, $C_{IBE,3} = M \cdot e(g, h_1)^r$, $\beta = H(C_{IBE,1}, C_{IBE,2}, C_{IBE,3})$, and $C_{IBE,4} = (e(g, h_2) e(g, h_3)^\beta)^r$, and output $C_{IBE} = (C_{IBE,1}, C_{IBE,2}, C_{IBE,3}, C_{IBE,4})$.*

$\mathsf{IBE.Dec}(s_{ID}, C_{IBE})$**:** *Parse $C_{IBE} = (C_{IBE,1}, C_{IBE,2}, C_{IBE,3}, C_{IBE,4})$. Compute $\beta = H(C_{IBE,1}, C_{IBE,2}, C_{IBE,3})$ and check*

$$e(C_{IBE,1}, h_{ID,2} h_{ID,3}^\beta) C_{IBE,2}^{r_{ID,2} + r_{ID,3}\beta} \overset{?}{=} C_{IBE,4}$$

If the chack fails, then output $\perp$. Otherwise, output $M = C_{IBE,3} / \{e(C_{IBE,1}, h_{ID,1}) \cdot C_{IBE,2}^{r_{ID,1}}\}$.

Due to the universal onewayness of H, it is hard to find $(C_{IBE,1}, C_{IBE,2}, C_{IBE,3})$ and $(C'_{IBE,1}, C'_{IBE,2}, C'_{IBE,3})$ such that $\beta = H(C_{IBE,1}, C_{IBE,2}, C_{IBE,3}) = H(C'_{IBE,1}, C'_{IBE,2}, C'_{IBE,3})$, and $(C_{IBE,1}, C_{IBE,2}, C_{IBE,3}) \neq (C'_{IBE,1}, C'_{IBE,2}, C'_{IBE,3})$. So, no adversary can issue a ciphertext to $\mathcal{DEC}$ with the condition that the hashed vallue of the ciphertext is the same as that of the challenge ciphertext (otherwise, we break the universal onewayness of H). This is an analogous on the Cramer-Shoup PKE [20].

Keita Emura received the B.E. and M.E. degrees from Kanazawa University, Ishikawa, Japan in 2002 and 2004, respectively. He worked at Fujitsu Hokuriku Systems Limited from 2004 to 2006, and was engaged in research and development for asynchronous communication. He received the Ph.D. degrees in information science from Japan Advanced Institute of Science and Technology (JAIST) in 2009. He has been a postdoctoral researcher at Center for Highly Dependable Embedded Systems Technology, JAIST since 2010. He received the Best Paper Award from ADMA in 2010. His research interests include cryptography and information security.

Atsuko Miyaji received the B.Sc., the M.Sc., and the Dr. Sci. degrees in mathematics from Osaka University, Osaka, Japan in 1988, 1990, and 1997 respectively. She joined Panasonic Co., LTD from 1990 to 1998 and engaged in research and development for secure communication. She was an associate professor at the Japan Advanced Institute of Science and Technology (JAIST) in 1998. She has joined the computer science department of the University of California, Davis since 2002. She has been a professor at the Japan Advanced Institute of Science and Technology (JAIST) since 2007 and the director of Library of JAIST since 2008. Her research interests include the application of number theory into cryptography and information security. She received the IPSJ Sakai Special Researcher Award in 2002, the Standardization Contribution Award in 2003, Engineering Sciences Society: Certificate of Appreciation in 2005, the AWARD for the contribution to CULTURE of SECURITY in 2007, IPSJ/ITSCJ Project Editor Award in 2007, 2008, 2009, and 2010, the Director-General of Industrial Science and Technology Policy and Environment Bureau Award in 2007, Editorial Committee of Engineering Sciences Society: Certificate of Appreciation in 2007, DoCoMo Mobile Science Awards in 2008, Advanced Data Mining and Applications (ADMA2010) Best Paper Award, and the chief of air staff: Letter of Appreciation Award. She is a member of the International Association for Cryptologic Research, the Information Processing Society of Japan, and the Mathematical Society of Japan.

Kazumasa Omote received his M.S. and Ph.D. degrees in information science from Japan Advanced Institute of Science and Technology (JAIST) in 1999 and 2002, respectively. He worked at Fujitsu Laboratories, LTD from 2002 to 2008, and was engaged in research and development for network security. He has been a research assistant professor at Japan Advanced Institute of Science and Technology (JAIST) since 2008 and has been an associate professor at JAIST since 2011. His research interests include applied cryptography and network security. He is a member of the IPS of Japan.