Timed-Release Certificateless Encryption

Toru Oshikiri
Graduate School of Engineering
Tokyo Denki University
Tokyo, Japan

Taiichi Saito
Tokyo Denki University
Tokyo, Japan

***Abstract*—Timed-Release Encryption(TRE) is an encryption mechanism that allows a receiver to decrypt a ciphertext only after the time that a sender designates. In this paper, we propose the notion of Timed-Release Certificateless Encryption(TRCLE), and define its security models. We also show a generic construction of TRCLE from Public-Key Encryption(PKE), Identity-Based Encryption(IBE) and one-time signature, and prove that the constructed scheme achieves the security we defined.**

***Keywords*—timed-release encryption, identity-based encryption, one-time signature**

I. Introduction

This paper introduces the notion of *Timed-Release Certificateless Encryption (TRCLE)*. TRCLE is a variant of *Timed-Release Encryption (TRE)* [1] [2], in which a sender can generate a ciphertext designating a time to decrypt it, and a receiver can decrypt the ciphertext only after the designated time.

A TRCLE system consists of a key generation center (KGC), a time server (TS), senders and receivers. The KGC helps a receiver to generate a decryption key corresponding to the ID and public key of a receiver. The TS periodically broadcasts a time signal corresponding to the current time. A sender encrypts a message using an ID and a public key of a receiver and a time after which the ciphertext could be decrypted. The receiver decrypts the ciphertext using the decryption key and the time signal corresponding to the time designated by the sender. The TRCLE system does not allow the KGC to obtain the decryption key of receiver, and then it allows only the receiver to decrypt the ciphertext only after the designated time.

The decryption key consists of two keys, a partial secret key and user secret key. Since the former is generated by the KGC and the user but the latter only by the user, the KGC does not know the whole decryption key and cannot decrypt ciphertext.

II. Application

TRCLE has an application to online "sealed-bid auction" in online community in which each registered user has an ID. In the auction system, every user can become auctioneer by publicizing his ID and public key. Each bidder encrypts his price by using the auctioneer's ID and public key and submits the ciphertext as sealed-bid. The auctioneer can decrypt all bids only after the pre-determined closing time. In the sealed-bid auction based on TRCLE, each user determines whether he trusts the auctioneer of ID and attends the auction by checking the reputations and the transaction records in the past auctions organized by the user of ID. Every user easily starts a sealed-bid auction based on TRCLE, since it does not requires heavy infrastructure linking public keys to ID such as Public-Key Infrastructure (PKI).

III. Related Works

There is another variant of TRE, *Timed-Release Identity-Based Encryption (TRIBE)* [3] [4]. In TRIBE, a user can decrypt a ciphertext only when the user has the receiver's secret key and the time signal generated by TS. Then, if the receiver does not have the time signal or the TS does not have the secret key, they cannot decrypt the ciphertext. In [3], two security models of TRIBE are defined. One is security against malicious receiver, $\mathsf{IND\text{-}ID\text{-}CCA}_{\mathsf{CR}}$ security. The other is security against malicious TS, $\mathsf{IND\text{-}ID\text{-}CCA}_{\mathsf{TS}}$ security. A generic construction of TRIBE that achieves the security is also shown in [3]. It is a combination of two IBE schemes and a one-time signature schemes, based on "Parallel Encryption" by Dodis-Katz [5], and the security is proved in the standard model.

TRCLE has an advantage over TRIBE in that a compromised KGC cannot decrypt any ciphertext since the key generation process is split between the KGC and the user. Then we discuss only security against malicious KGC in this paper. The other security is proved in almost the same way as in TRIBE.

TRCLE can be considered also a variant of *Certificateless Encryption (CLE)* [6] having the mechanism of TRE. In CLE, the decryption key is partially determined by KGC

IV. Contributions

In this paper, we introduce timed-release certificateless encryption (TRCLE) and define its security models including security against malicious KGC, $\mathsf{Mal.KGC}$ security. We also present a generic construction of TRCLE. It is a combination of a Public-Key Encryption(PKE) scheme, two Identity-Based Encryption(IBE) schemes and a one-time signature schemes, also based on "Parallel Encryption". We see that if the primitive PKE scheme is *indistinguishability against adaptive chosen ciphertext attacks*($\mathsf{IND\text{-}CCA}$) secure and the primitive one-time signature scheme is *one-time strong existential unforgeability against chosen message attacks*($\mathsf{OT\text{-}sEUF\text{-}CMA}$) secure, then the constructed TRCLE scheme is $\mathsf{Mal.KGC}$ secure in the standard model.

V. Preliminaries

In this section, we review public-key encryption (PKE), identity-based encryption (IBE) and one-time signature, which we use later.

A. Public-Key Encryption

Let λ be a security parameter. An *public-key encryption scheme* $\mathcal{PKE}$ [7] consists of three probabilistic polynomial-time algorithms $\mathcal{PKE}$ = ($\mathsf{PKE.Gen}$, $\mathsf{PKE.Enc}$, $\mathsf{PKE.Dec}$). The key generation algorithm $\mathsf{PKE.Gen}$ takes λ as input, and outputs a public key pk and a secret key sk. The encryption algorithm $\mathsf{PKE.Enc}$ takes pk, a message m as inputs, and outputs a ciphertext c. The decryption algorithm $\mathsf{PKE.Dec}$ takes a secret key sk and a ciphertext c as inputs, and outputs the plaintext m' or $\perp$. These algorithms are assumed to satisfy that if (pk, sk) = $\mathsf{PKE.Gen}(\lambda)$ then $\mathsf{PKE.Dec}(sk, \mathsf{PKE.Enc}(pk, m)) = m$ for any m.

1) IND-CCA *Security:* We review a standard security notion for PKE: *indistinguishability against adaptive chosen ciphertext attacks (*IND-CCA*)* security. We here describe the IND-CCA security for PKE scheme $\mathcal{PKE}$ based on the following IND-CCA game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup
$\mathcal{C}$ runs $(pk, sk) \leftarrow \mathsf{PKE.Gen}(\lambda)$. $\mathcal{C}$ sends pk to $\mathcal{A}$ and keeps sk secret.

Phase1
$\mathcal{A}$ can adaptively issue *decryption queries* c. $\mathcal{C}$ responds to a decryption query c by running $m' = \mathsf{PKE.Dec}(sk, c)$, and returning m' to $\mathcal{A}$.

Challenge
$\mathcal{A}$ sends two messages m_0, m_1 such that $|m_0| = |m_1|$ to $\mathcal{C}$. $\mathcal{C}$ randomly chooses $b \in \{0, 1\}$ and sends a challenge ciphertext $c^* = \mathsf{PKE.Enc}(pk, m_b)$ to $\mathcal{A}$.

Phase2
$\mathcal{A}$ can adaptively issue *decryption queries* c in the same way as in **Phase1** except that the decryption queries c must differ from the challenge ciphertext c^*.

Guess
$\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins if $b = b'$.

We define an advantage of $\mathcal{A}$ in the IND-CCA game as $Adv^{\mathsf{IND\text{-}CCA}}_{\mathcal{PKE},\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$, in which the probability is taken over the random coins used by $\mathcal{C}$ and $\mathcal{A}$. We say that the PKE scheme $\mathcal{PKE}$ is IND-CCA *secure* if, for any probabilistic polynomial-time adversary $\mathcal{A}$, the function $Adv^{\mathsf{IND\text{-}CCA}}_{\mathcal{PKE},\mathcal{A}}(\lambda)$ is negligible in λ.

B. Identity-Based Encryption

Let λ be a security parameter. An *identity-based encryption scheme* $\mathcal{IBE}$ [8] consists of four probabilistic polynomial-time algorithms $\mathcal{IBE}$ = ($\mathsf{IBE.Setup}$, $\mathsf{IBE.Ext}$, $\mathsf{IBE.Enc}$, $\mathsf{IBE.Dec}$). The setup algorithm $\mathsf{IBE.Setup}$ takes λ as input, and outputs a public parameter $params$ and a master secret key msk. The extract algorithm $\mathsf{IBE.Ext}$ takes $params$, msk, and an identity ID as inputs, and outputs a decryption key d_{ID}. The encryption algorithm $\mathsf{IBE.Enc}$ takes $params, \mathsf{ID}$, a message m as inputs, and outputs a ciphertext c. The decryption algorithm $\mathsf{IBE.Dec}$ takes $params$, a decryption key d_{ID} and a ciphertext c as inputs, and outputs the plaintext m' or $\perp$. These algorithms are assumed to satisfy that if $(params, msk) = \mathsf{IBE.Setup}(\lambda)$ and $d_{\mathsf{ID}} = \mathsf{IBE.Ext}(params, msk, \mathsf{ID})$ then $\mathsf{IBE.Dec}(params, d_{\mathsf{ID}}, \mathsf{IBE.Enc}(params, \mathsf{ID}, m)) = m$ for any m.

1) IND-ID-CCA *Security:* We review a standard security notion for IBE: *indistinguishability against adaptive identity and chosen ciphertext attacks (*IND-ID-CCA*)* security [9]. We here describe the IND-ID-CCA security for IBE scheme $\mathcal{IBE}$ based on the following IND-ID-CCA game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup
$\mathcal{C}$ runs $(params, msk) \leftarrow \mathsf{IBE.Setup}(\lambda)$. $\mathcal{C}$ sends $params$ to $\mathcal{A}$ and keeps msk secret.

Phase1
$\mathcal{A}$ can adaptively issue *extraction queries* ID and *decryption queries* (ID, c). $\mathcal{C}$ responds to an extraction query ID by running d_{ID} = $\mathsf{IBE.Ext}(params, msk, \mathsf{ID})$ and returning d_{ID} to $\mathcal{A}$. $\mathcal{C}$ responds to a decryption query (ID, c) by running $d_{\mathsf{ID}} = \mathsf{IBE.Ext}(params, msk, \mathsf{ID})$ and $m' = \mathsf{IBE.Dec}(params, d_{\mathsf{ID}}, c)$, and returning m' to $\mathcal{A}$.

Challenge
$\mathcal{A}$ sends two messages m_0, m_1 such that $|m_0| = |m_1|$, and an identity to be challenged ID^* to $\mathcal{C}$. The challenge identity ID^* must differ from any ID issued as extraction query in **Phase1**. $\mathcal{C}$ randomly chooses $b \in \{0, 1\}$ and sends a challenge ciphertext $c^* = \mathsf{IBE.Enc}(params, \mathsf{ID}^*, m_b)$ to $\mathcal{A}$.

Phase2
$\mathcal{A}$ can adaptively issue extraction queries ID and decryption queries (ID, c) in the same way as in **Phase1** except that the extraction queries ID must differ from the challenge identity ID^*, and decryption queries (ID, c) must differ from the pair (ID^*, c^*).

Guess
$\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins if $b = b'$.

We define an advantage of $\mathcal{A}$ in the IND-ID-CCA game as $Adv^{\mathsf{IND\text{-}ID\text{-}CCA}}_{\mathcal{IBE},\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$, in which the probability is taken over the random coins used by $\mathcal{C}$ and $\mathcal{A}$. We say that the IBE scheme $\mathcal{IBE}$ is IND-ID-CCA *secure* if, for any probabilistic polynomial-time adversary $\mathcal{A}$, the function $Adv^{\mathsf{IND\text{-}ID\text{-}CCA}}_{\mathcal{IBE},\mathcal{A}}(\lambda)$ is negligible in λ.

C. One-time Signature

Let λ be a security parameter. $\mathcal{A}$ *signature* scheme $\mathcal{SIG}$ consists of three probabilistic polynomial-time algorithms $\mathcal{SIG}$ = $(\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$. The key generation algorithm SigGen takes λ as input, and outputs a signing key sk and a verification key vk. The signing algorithm Sign takes sk and a message m as inputs , and outputs a signature σ. The verification algorithm Verify takes vk, a message m, and a signature σ as inputs, and outputs `accept` or `reject`. These algorithms are assumed to satisfy that if $(sk, vk) = \mathsf{SigGen}(\lambda)$ then $\mathsf{Verify}(vk, m, \mathsf{Sign}(sk, m)) = \texttt{accept}$ for any m.

1) OT-sEUF-CMA *Security:* We review a security notion for one-time signature scheme: *one-time strong existential unforgeability against chosen message attacks (*OT-sEUF-CMA*)* security [10]. We here describe the OT-sEUF-CMA security for signature scheme $\mathcal{SIG}$ based on the following OT-sEUF-CMA game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup
$\mathcal{C}$ runs the $(sk, vk) \leftarrow \mathsf{SigGen}(\lambda)$. $\mathcal{C}$ sends vk to $\mathcal{A}$ and keeps sk secret.

Query
$\mathcal{A}$ can issue a *signing query* m to $\mathcal{C}$ only once. $\mathcal{C}$ responds to the singing query m by running $\sigma = \mathsf{Sign}(vk, m)$ and returning σ to $\mathcal{A}$.

Forge
$\mathcal{A}$ outputs a pair (m^*, σ^*).

We define the advantage of $\mathcal{A}$ in the OT-sEUF-CMA game as $Adv^{\text{OT-sEUF-CMA}}_{\mathcal{SIG},\mathcal{A}}(\lambda) = \Pr[\mathsf{Verify}(vk,\, m^*,\, \sigma^*) = \texttt{accept} \wedge (m, \sigma) \neq (m^*, \sigma^*)]$, in which the probability is taken over the random coins used by $\mathcal{C}$ and $\mathcal{A}$. We say that the signature scheme $\mathcal{SIG}$ is OT-sEUF-CMA *secure* if, for any probabilistic polynomial-time adversary $\mathcal{A}$, the function $Adv^{\text{OT-sEUF-CMA}}_{\mathcal{SIG},\mathcal{A}}(\lambda)$ is negligible in λ.

VI. Timed-Release Certificateless Encryption(TRCLE)

In this section, we introduce timed-release certificateless encryption(TRCLE) scheme and define its security models.

A TRCLE system consists of a key generation center (KGC), a time server (TS), senders and receivers. The KGC helps a receiver to generate a partial secret key corresponding to an ID of the receiver. The TS periodically broadcasts a time signal corresponding to the current time. A sender encrypts a message using the ID and public key of a receiver, and a time after which the ciphertext could be decrypted. The receiver decrypts the ciphertext using the partial secret key, the user secret key and the time signal corresponding to the time designated by the sender.

Let λ be a security parameter. An *timed-release certificateless encryption scheme* $\mathcal{TRCLE}$ consists of seven probabilistic polynomial-time algorithms $\mathcal{TRCLE}$ = (KGC_Setup, TS_Setup, PartialKeyGen, UserKeyGen, Release, Encrypt, Decrypt). The key generation center's setup algorithm KGC_Setup takes λ as input, and outputs a public parameter $params$ and a master secret key msk. The time server's setup algorithm TS_Setup takes λ as input, and outputs a public key tpk and the corresponding secret key tsk. The partial secret key generation algorithm PartialKeyGen takes $params, msk$ and ID as input, and outputs a partial secret key psk_{ID} corresponding to ID. The user key generation algorithm UserKeyGen takes $params$ and ID as input, and outputs a user public key upk_{ID} and a user secret key usk_{ID} corresponding to ID. The release algorithm Release takes tpk, tsk and a time period T as inputs, and outputs a time signal d_T. The encryption algorithm Encrypt takes $params$, tpk, ID, upk_{ID}, T and a message m as inputs, and outputs a ciphertext c. The decryption algorithm Decrypt takes as inputs $params$, tpk, psk_{ID}, usk_{ID}, d_T and a ciphertext c', and outputs the plaintext m' or $\perp$. These algorithms are assumed to satisfy that $\mathsf{Decrypt}(params, tpk, psk_{\mathsf{ID}}, usk_{\mathsf{ID}}, d_T, \mathsf{Encrypt}(params, tpk, \mathsf{ID}, upk_{\mathsf{ID}}, T, m)) = m$ holds for any m, if $(tpk, tsk) = \mathsf{TS\_Setup}(\lambda)$, $(params, msk) = \mathsf{KGC\_Setup}(\lambda)$, $psk_{\mathsf{ID}} = \mathsf{PartialKeyGen}(params, msk, \mathsf{ID})$, $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) = \mathsf{UserKeyGen}(params, \mathsf{ID})$ and $d_T = \mathsf{TR.Release}(tpk,\, tsk,\, T)$ hold.

A. Security

We can consider security against KGC, TS, receiver or outsider. Since the TS security is implied by KGC security, we present the three kinds of security.

1) Mal.KGC *Security:* We introduce a security notion for TRCLE: *indistinguishability against adaptive identity and chosen ciphertext attacks by key generation center (*Mal.KGC*) security*. This security ensures that a malicious key generation center, who has a master secret key msk, cannot obtain any information of message from a ciphertext without a user secret key usk_{ID}. We here describe the Mal.KGC security for a TRCLE scheme $\mathcal{TRCLE}$ based on the following Mal.KGC game between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

Setup
$\mathcal{C}$ runs $(tpk, tsk) \leftarrow \mathsf{TS\_Setup}(\lambda)$ and sends (λ, tpk, tsk) to $\mathcal{A}$. $\mathcal{A}$ runs $(params, msk) \leftarrow \mathsf{KGC\_Setup}(\lambda)$ and sends $params$ to $\mathcal{C}$. $\mathcal{C}$ creates an empty list $\texttt{List}$.

Phase1
$\mathcal{A}$ can adaptively issue the following four queries.

Create User query
$\mathcal{A}$ sends $(\mathsf{ID}, psk_{\mathsf{ID}})$ to $\mathcal{C}$. When ID is in $\texttt{List}$, $\mathcal{C}$ returns upk_{ID} corresponding to ID. When ID is not in $\texttt{List}$, $\mathcal{C}$ runs $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) \leftarrow \mathsf{UserKeyGen}(params, \mathsf{ID})$ and stores $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ in $\texttt{List}$. $\mathcal{C}$ returns upk_{ID} corresponding to ID.

Reveal Secret Key query
$\mathcal{A}$ sends ID to $\mathcal{C}$. When ID is in $\texttt{List}$, $\mathcal{C}$ returns usk_{ID} corresponding to ID. When ID is not in $\texttt{List}$, $\mathcal{C}$ returns $\perp$

Replace query
$\mathcal{A}$ sends $(\mathsf{ID}, upk', usk')$ to $\mathcal{C}$. When ID is in $\texttt{List}$, $\mathcal{C}$ replaces $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ with $(\mathsf{ID}, psk_{\mathsf{ID}}, upk', usk')$. If $usk' = \perp$, $\mathcal{C}$ sets $usk' = usk_{\mathsf{ID}}$ When ID is not in $\texttt{List}$, $\mathcal{C}$ do nothing.

Decrypt query
$\mathcal{A}$ sends (ID, T, c) to $\mathcal{C}$. When ID is in $\texttt{List}$, $\mathcal{C}$ runs $d_T \leftarrow \mathsf{Release}(tpk, tsk, T)$ and $m \leftarrow \mathsf{Decrypt}(params, tpk, psk_{\mathsf{ID}}, usk_{\mathsf{ID}}, d_T, c)$ and returns m. When ID is not in $\texttt{List}$, $\mathcal{C}$ returns $\perp$.

Challenge
$\mathcal{A}$ sends two messages m_0, m_1 such that $|m_0| = |m_1|$, an identity to be challenged ID^* and a time period T^* to $\mathcal{C}$. The challenge identity ID^* must differ from any ID issued as Replace

queries in **Phase1**. $\mathcal{C}$ randomly chooses $b \in \{0,1\}$ and sends a challenge ciphertext $c^* = \mathsf{Encrypt}(params, tpk, \mathsf{ID}^*, upk_{\mathsf{ID}^*}, T^*, m_b)$ to $\mathcal{A}$.

Phase2

$\mathcal{A}$ can adaptively issue the above four queries in the same way as **Phase1** except that the Replace queries ID must differ from the challenge identity ID^*, and the decryption queries (ID, T, c) must differ from the tuple $(\mathsf{ID}^*, T^*, c^*)$.

Guess

$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins if $b = b'$.

We define an advantage of $\mathcal{A}$ in the Mal.KGC game as $Adv^{\mathsf{Mal.KGC}}_{\mathcal{TRCLE},\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$, in which the probability is taken over the random coins used by $\mathcal{C}$ and $\mathcal{A}$. We say that the TRCLE scheme $\mathcal{TRCLE}$ is Mal.KGC *secure* if, for any probabilistic polynomial-time adversary $\mathcal{A}$, the function $Adv^{\mathsf{Mal.KGC}}_{\mathcal{TRCLE},\mathcal{A}}(\lambda)$ is negligible in λ.

2) Mal.Receiver *Security:* We introduce a security notion for TRCLE: *indistinguishability against adaptive identity and chosen ciphertext attacks by receiver (*Mal.Receiver*) security*. This security ensures that a malicious receiver, who has a partial secret key psk_{ID} and user secret key usk_{ID}, cannot obtain any information of message from a ciphertext without a time signal d_T corresponding to the time designated by the sender. We here describe the Mal.Receiver security for a TRCLE scheme $\mathcal{TRCLE}$ based on the following Mal.Receiver game between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

Setup

$\mathcal{C}$ runs (*params, msk*) $\leftarrow$ KGC_Setup(λ), (*tpk, tsk*) $\leftarrow$ TS_Setup(λ), and sends (λ, tpk, tsk) to $\mathcal{A}$. $\mathcal{C}$ creates an empty list `List`.

Phase1

$\mathcal{A}$ can adaptively issue the following five queries.

Create User query

$\mathcal{A}$ sends (ID, upk_{ID}, usk_{ID}) to $\mathcal{C}$. When ID is in `List`, $\mathcal{C}$ do nothing. When ID is not in `List`, $\mathcal{C}$ runs $psk_{\mathsf{ID}} \leftarrow \mathsf{PartialKeyGen}(params, msk, \mathsf{ID})$ and stores $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ in `List`.

Reveal Partial Key query

$\mathcal{A}$ sends ID to $\mathcal{C}$. When ID is in `List`, $\mathcal{C}$ returns psk_{ID} corresponding to ID. When ID is not in `List`, $\mathcal{C}$ returns $\perp$

Replace query

$\mathcal{A}$ sends $(\mathsf{ID}, upk', usk')$ to $\mathcal{C}$. When ID is in `List`, $\mathcal{C}$ replaces $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ with $(\mathsf{ID}, psk_{\mathsf{ID}}, upk', usk')$. If $usk' = \perp$, $\mathcal{C}$ sets $usk' = usk_{\mathsf{ID}}$ When ID is not in `List`, $\mathcal{C}$ do nothing.

Release query

$\mathcal{A}$ sends T to $\mathcal{C}$. $\mathcal{C}$ runs $d_T \leftarrow$ Release(tpk, tsk, T) and returns d_T.

Decrypt query

$\mathcal{A}$ sends (ID, T, c) to $\mathcal{C}$. When ID is in `List`, $\mathcal{C}$ runs $d_T \leftarrow$ Release(tpk, tsk, T) and $m \leftarrow$ Decrypt($params$, tpk, psk_{ID}, usk_{ID}, d_T, c) and returns m. When ID is not in `List`, $\mathcal{C}$ returns $\perp$.

Challenge

$\mathcal{A}$ sends two messages m_0, m_1 such that $|m_0| = |m_1|$, an identity to be challenged ID^* and a time period T^* to $\mathcal{C}$. The time period T^* must differ from any T issued as Release queries in **Phase1**. $\mathcal{C}$ randomly chooses $b \in \{0,1\}$ and sends a challenge ciphertext $c^* = \mathsf{Encrypt}(params, tpk, \mathsf{ID}^*, upk_{\mathsf{ID}^*}, T^*, m_b)$ to $\mathcal{A}$.

Phase2

$\mathcal{A}$ can adaptively issue the above five queries in the same way as **Phase1** except that the Release queries T must differ from T^*, and the decryption queries (ID, T, c) must differ from the tuple $(\mathsf{ID}^*, T^*, c^*)$.

Guess

$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins if $b = b'$.

We define an advantage of $\mathcal{A}$ in the Mal.Receiver game as $Adv^{\mathsf{Mal.Receiver}}_{\mathcal{TRCLE},\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$, in which the probability is taken over the random coins used by $\mathcal{C}$ and $\mathcal{A}$. We say that the TRCLE scheme $\mathcal{TRCLE}$ is Mal.Receiver *secure* if, for any probabilistic polynomial-time adversary $\mathcal{A}$, the function $Adv^{\mathsf{Mal.Receiver}}_{\mathcal{TRCLE},\mathcal{A}}(\lambda)$ is negligible in λ.

3) Outsider *Security:* We introduce a security notion for TRCLE: *indistinguishability against adaptive identity and chosen ciphertext attacks by outsider (*Outsider*) security*. This security ensures that a outsider, who has a public parameter $params$ and tpk, cannot obtain any information of message from a ciphertext without user secret key usk_{ID}. We here describe the Outsider security for a TRCLE scheme $\mathcal{TRCLE}$ based on the following Outsider game between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

Setup

$\mathcal{C}$ runs (*params, msk*) $\leftarrow$ KGC_Setup(λ), (*tpk, tsk*) $\leftarrow$ TS_Setup(λ), and sends $(params, tpk)$ to $\mathcal{A}$. $\mathcal{C}$ creates an empty list `List`.

Phase1

$\mathcal{A}$ can adaptively issue the following six queries.

Create User query

$\mathcal{A}$ sends ID to $\mathcal{C}$. When ID is in `List`, $\mathcal{C}$ returns upk_{ID} corresponding to ID. When ID is not in `List`, $\mathcal{C}$ runs $psk_{\mathsf{ID}} \leftarrow$ PartialKeyGen($params$, msk, ID) and (upk_{ID}, usk_{ID}) $\leftarrow$ UserKeyGen($params$, ID), and stores (ID, psk_{ID}, upk_{ID}, usk_{ID}) in `List`. $\mathcal{C}$ returns upk_{ID} corresponding to ID.

Reveal Partial Key query

$\mathcal{A}$ sends ID to $\mathcal{C}$. When ID is in `List`, $\mathcal{C}$ returns psk_{ID} corresponding to ID. When ID is not in `List`, $\mathcal{C}$ returns $\perp$

Reveal Secret Key query

$\mathcal{A}$ sends ID to $\mathcal{C}$. When ID is in `List`, $\mathcal{C}$ returns usk_{ID} corresponding to ID. When ID is not in `List`, $\mathcal{C}$ returns $\perp$

Replace query

$\mathcal{A}$ sends $(\mathsf{ID}, upk', usk')$ to $\mathcal{C}$. When ID is in `List`, $\mathcal{C}$ replaces (ID, psk_{ID}, upk_{ID}, usk_{ID}) with (ID, psk_{ID}, upk', usk'). If $usk' = \perp$, $\mathcal{C}$ sets $usk' = usk_{\mathsf{ID}}$ When ID is not in `List`, $\mathcal{C}$ do nothing.

Release query

$\mathcal{A}$ sends T to $\mathcal{C}$. $\mathcal{C}$ runs $d_T \leftarrow$ Release(tpk, tsk, T) and returns d_T.

Decrypt query

$\mathcal{A}$ sends (ID, T, c) to $\mathcal{C}$. When ID is in `List`, $\mathcal{C}$ runs $d_T \leftarrow$ Release(tpk, tsk, T) and $m \leftarrow$ Decrypt($params, tpk, psk_{\mathsf{ID}}, usk_{\mathsf{ID}}, d_T, c$) and returns m. When ID is not in `List`, $\mathcal{C}$ returns $\perp$.

Challenge

$\mathcal{A}$ sends two messages m_0, m_1 such that $|m_0| = |m_1|$, an identity to be challenged ID* and a time period T^* to $\mathcal{C}$. The challenge identity ID* must differ from any ID issued as Reveal Partial Key queries in **Phase1**. $\mathcal{C}$ randomly chooses $b \in \{0,1\}$ and sends a challenge ciphertext $c^* =$ Encrypt($params, tpk, \mathsf{ID}^*, upk_{\mathsf{ID}^*}, T^*, m_b$) to $\mathcal{A}$.

Phase2

$\mathcal{A}$ can adaptively issue the above six queries in the same way as **Phase1** except that the Reveal Secret Key queries ID must differ from ID*, and the decryption queries (ID, T, c) must differ from the tuple (ID*, T^*, c^*).

Guess

$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins if $b = b'$.

We define an advantage of $\mathcal{A}$ in the Outsider game as $Adv^{\mathsf{Outsider}}_{\mathcal{TRCLE},\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$, in which the probability is taken over the random coins used by $\mathcal{C}$ and $\mathcal{A}$. We say that the TRCLE scheme $\mathcal{TRCLE}$ is Outsider *secure* if, for any probabilistic polynomial-time adversary $\mathcal{A}$, the function $Adv^{\mathsf{Outsider}}_{\mathcal{TRCLE},\mathcal{A}}(\lambda)$ is negligible in λ.

VII. Construction

Here we present a generic construction of TRCLE scheme from PKE scheme, IBE scheme and one-time signature scheme.

A. Construction

Let $\Delta =$ (PKE.Gen, PKE.Enc, PKE.Dec) be a public-key encryption scheme, $\Pi =$ (IBE.Setup, IBE.Ext, IBE.Enc, IBE.Dec) and $\Pi' =$ (IBE$'$.Setup, IBE$'$.Ext, IBE$'$.Enc, IBE$'$.Dec) be identity-based encryption schemes, and $\Sigma =$ (SigGen, Sign, Verify) be a one-time signature scheme.

A TRCLE scheme $\Gamma =$ (KGC_Setup, TS_Setup, PartialKeyGen, UserKeyGen, Release, Encrypt, Decrypt) is constructed as follows.

KGC_Setup(λ):
Step 1: Run IBE.Setup on input λ to generate $(params, msk)$.
Step 2: Return $(params, msk)$.

TS_Setup(λ):
Step 1: Run IBE$'$.Setup on input λ to generate $(params', msk')$.
Step 2: Set $tpk = params'$ and $tsk = msk'$.
Step 3: Return (tpk, tsk).

PartialKeyGen($params, msk, ID$):
Step 1: Run IBE.Ext($params, msk, ID$) to obtain d_{ID}.
Step 2: Return d_{ID}.

UserKeyGen($params, ID$):
Step 1: Run PKE.Gen on input λ to generate (pk, sk).
Step 2: Set $upk_{ID} = pk$ and $usk_{\mathsf{ID}} = sk$.
Step 3: Return (upk_{ID}, usk_{ID}).

Release(tpk, tsk, T):
Step 1: Run IBE$'$.Ext(tpk, tsk, T) to obtain d_T.
Step 2: Return d_T.

Encrypt($params, tpk, ID, upk_{ID}, T, m$):
Step 1: Run SigGen on input λ to generate (sk, vk).
Step 2: Randomly choose $s_1 \in \{0,1\}^{|m|}$ and $s_2 \in \{0,1\}^{|m|}$.
Step 3: Compute $s_3 = s_1 \oplus s_2 \oplus m$.
Step 4: Compute c_1=IBE.Enc($params$, ID, $s_1||vk$).
Step 5: Compute c_2=IBE$'$.Enc(tpk, $T, s_2||vk$).
Step 6: Compute c_3=PKE.Enc($upk_{ID}, s_3||vk$).
Step 7: Compute $\sigma =$ Sign($sk, c_1||c_2||c_3||ID||T$).
Step 8: Set $c = (c_1, c_2, c_3, ID, T, vk, \sigma)$.
Step 9: Return c.

Decrypt($params, tpk, psk_{ID}, usk_{ID}, d_T, c$):
Step 1: Parse c as $c = (c_1, c_2, c_3, ID, T, vk, \sigma)$.
Step 2: If Verify($vk, c_1||c_2||c_3||ID||T, \sigma$) = `reject` , then return $\perp$ and stop.
Step 3: Compute $s_1||vk'$ = IBE.Dec($params$, psk_{ID}, c_1).
Step 4: Compute $s_2||vk''$ = IBE$'$.Dec(tpk, d_T, c_2).
Step 5: Compute $s_3||vk'''$ = PKE.Dec(usk_{ID}, c_3).
Step 6: If $vk = vk' = vk'' = vk'''$, then return $m = s_1 \oplus s_2 \oplus s_3$ else return $\perp$.

B. Security of TRCLE

1) Mal.KGC *security:*

Theorem 1: If Δ is an IND-CCA secure public-key encryption scheme and Σ is an OT-sEUF-CMA secure one-time signature scheme, then Γ is a Mal.KGC secure timed-release certificateless encryption scheme.

Proof(Theorem 1) Suppose $\mathcal{A}$ is an adversary that breaks the Mal.KGC security of Γ. We construct a simulator $\mathcal{B}$ which breaks the IND-CCA security of the PKE scheme Δ using $\mathcal{A}$. Here we say a ciphertext $c = (c_1, c_2, c_3, \mathsf{ID}, T, vk, \sigma)$ is *valid* if Verify($vk, c_1||c_2||c_3||\mathsf{ID}||T, \sigma$) $=$ `accept`. Let $c^* = (c_1^*, c_2^*, c_3^*, \mathsf{ID}^*, T^*, vk^*, \sigma^*)$ be the challenge ciphertext. Let `Forge` denote the event that $\mathcal{A}$ submits a valid ciphertext $c = (c_1, c_2, c_3, T, \mathsf{ID}, vk^*, \sigma)$ as a Decrypt query to $\mathcal{C}$ in the **Phase2**, and `Succ` denote the event that $\mathcal{A}$ wins the Mal.KGC game. We assume that $\mathcal{A}$ issues at most q distinct Create User queries. We prove the following claims.

Claim 1: $\Pr[\texttt{Forge}]$ is negligible.

Claim 2: $|\Pr[\texttt{Succ} \wedge \overline{\texttt{Forge}}] + \frac{1}{2}\Pr[\texttt{Forge}] - \frac{1}{2}|$ is negligible.

Proof(Claim 1) We assume `Forge` occurs. Then, we construct a forger $\mathcal{F}$ who breaks OT-sEUF-CMA security of the one-time signature scheme Σ, from $\mathcal{A}$. The description of $\mathcal{F}$ is as follows.

Setup

$\mathcal{F}$ receives vk^* from $\mathcal{C}$. $\mathcal{F}$ runs (tpk, tsk) $\leftarrow$ TS_Setup(λ) and sends (λ, tpk, tsk) to $\mathcal{A}$.

Query

$\mathcal{F}$ responds $\mathcal{A}$'s four queries as the challenger in the Mal.KGC game. If $\mathcal{A}$ happens to issue a valid ciphertext $c = (c_1, c_2, c_3, \mathsf{ID}, T, vk^*, \sigma)$

as decryption query to $\mathcal{F}$ before **Challenge** in the Mal.KGC game, then $\mathcal{F}$ simply outputs $(c_1||c_2||c_3||T||\mathsf{ID}, \sigma)$ as forgery and stops.

Challenge

If $\mathcal{A}$ outputs $(m_0, m_1, \mathsf{ID}^*, T^*)$ as challenge , $\mathcal{F}$ randomly chooses $s_1 \in \{0,1\}^{|m|}$, $s_2 \in \{0,1\}^{|m|}$ and $b \in \{0,1\}$, and computes $s_3 = s_1 \oplus s_2 \oplus m_b$. Then $\mathcal{F}$ computes $c_1^* = \mathsf{IBE.Enc}(params, \mathsf{ID}^*, s_1||vk^*)$, $c_2^* = \mathsf{IBE}'.\mathsf{Enc}(tpk, T^*, s_2||vk^*)$ and $c_3^* = \mathsf{PKE.Enc}(pub_{\mathsf{ID}^*}, s_3||vk^*)$, then issues $m^* = (c_1||c_2||c_3||\mathsf{ID}^*||T^*)$ as signing query to $\mathcal{C}$ and obtains σ^*. Finally $\mathcal{F}$ returns $c^* = (c_1^*, c_2^*, c_3^*, \mathsf{ID}^*, T^*, vk^*, \sigma^*)$ as the challenge ciphertext to $\mathcal{A}$.

Forge

If $\mathcal{A}$ issues a valid ciphertxt $c = (c_1, c_2, c_3, \mathsf{ID}, T, vk^*, \sigma)$ as decryption query, then $\mathcal{F}$ outputs $(c_1||c_2||c_3||\mathsf{ID}||T, \sigma)$ as forgery.

$\mathcal{F}$ can forge the signature if $\mathcal{A}$ issues a decryption query that causes the event $\mathtt{Forge}$. It, however, contradicts that Σ is OT-sEUF-CMA secure. Thus, $\Pr[\mathtt{Forge}]$ is negligible. □

Proof(Claim 2) We construct an adversary $\mathcal{B}$ who breaks IND-CCA security of the PKE scheme Δ using $\mathcal{A}$. The description of $\mathcal{B}$ is as follows.

Setup

$\mathcal{B}$ receives pk^* from $\mathcal{C}$. Then $\mathcal{B}$ runs $(tpk, tsk) \leftarrow \mathsf{TS\_Setup}(\lambda)$ and sends (λ, tpk, tsk) to $\mathcal{A}$ and randomly chooses index $i \in \{1, 2, \cdots q\}$. $\mathcal{B}$ creates an empty list $\mathtt{List}$.

Phase1

The response of $\mathcal{B}$ for $\mathcal{A}$'s queries is as follows.

Create User

When a given query $(\mathsf{ID}, psk_{\mathsf{ID}})$ is the i-th Create User query, $\mathcal{B}$ stores $(ID, psk_{\mathsf{ID}}, pk^*, \perp)$ into $\mathtt{List}$ and returns pk^* as upk_{ID}. When it is not the i-th, $\mathcal{B}$ runs $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) \leftarrow \mathsf{UserKeyGen}(params, \mathsf{ID})$. Then $\mathcal{B}$ stores $(ID, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ into $\mathtt{List}$ and returns upk_{ID}.

Reveal Secret Key

When ID is in $\mathtt{List}$, if $upk_{\mathsf{ID}} = pk^*$, $\mathcal{B}$ stops and outputs random bit b' and otherwise returns usk_{ID}. When ID is not in $\mathtt{List}$, $\mathcal{B}$ returns $\perp$.

Replace

When ID is in $\mathtt{List}$, if $upk_{\mathsf{ID}} = pk^*$, $\mathcal{B}$ stops and outputs random bit b'. If $usk' = \perp$, $\mathcal{B}$ sets $usk' = usk_{\mathsf{ID}}$.Then it replaces $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ with $(\mathsf{ID}, psk_{\mathsf{ID}}, upk', usk')$.

Decrypt

When ID is in $\mathtt{List}$, if $upk_{\mathsf{ID}} \neq pk^*$, $\mathcal{B}$ runs $d_T \leftarrow \mathsf{Release}(tpk, tsk, T)$ and $m \leftarrow \mathsf{Decrypt}(params, tpk, psk_{\mathsf{ID}}, usk_{\mathsf{ID}}, d_T, c)$ and returns m. If $upk_{\mathsf{ID}} = pk^*$, $\mathcal{B}$ responds as follows. If $\mathsf{Verify}(vk, c_1||c_2||c_3||\mathsf{ID}||T, \sigma) = \mathtt{reject}$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. Otherwise $\mathcal{B}$ runs $s_1||vk' \leftarrow \mathsf{IBE.Dec}(params, d_{\mathsf{ID}}, c_1)$ $s_2||vk'' \leftarrow \mathsf{IBE}'.\mathsf{Dec}(tpk, d_T, c_2,)$ and issues decryption query c_3 to $\mathcal{C}$ and obtains $s_3||vk'''$. $\mathcal{B}$ returns $m = s_1 \oplus s_2 \oplus s_3$ to $\mathcal{A}$ if $vk = vk' = vk''$, and otherwise $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. When ID is not in $\mathtt{List}$, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.

Challenge

If $\mathcal{A}$ outputs $(m_0, m_1, \mathsf{ID}^*, T^*)$ as challenge, $\mathcal{B}$ runs $(sk^*, vk^*) \leftarrow \mathsf{SigGen}(\lambda)$ and randomly chooses $s_1 \in \{0,1\}^{|m|}$ and $s_2 \in \{0,1\}^{|m|}$ and runs $c_1^* = \mathsf{IBE.Enc}(params, \mathsf{ID}^*, s_1||vk^*)$ and $c_2^* = \mathsf{IBE}'.\mathsf{Enc}(tpk, T^*, s_2||vk^*)$. Then $\mathcal{B}$ computes $M_0 = [(s_1 \oplus s_2 \oplus m_0)||vk^*]$ and $M_1 = [(s_1 \oplus s_2 \oplus m_1)||vk^*]$, and issues (M_0, M_1) as $\mathcal{B}$'s challenge to $\mathcal{C}$ and obtains a ciphertext c_3^*. $\mathcal{B}$ runs $\sigma^* = \mathsf{Sign}(sk^*, c_1^*||c_2^*||c_3^*||\mathsf{ID}^*||T^*)$ and returns $c^* = (c_1^*, c_2^*, c_3^*, \mathsf{ID}^*, T^*, vk^*, \sigma^*)$ as challenge ciphertext to $\mathcal{A}$.

Phase2

$\mathcal{B}$ responds to Create User query, Reveal Secret Key query and Replace query in the same way as in **Phase1**. $\mathcal{B}$ responds to Decrypt query as follows.The followings are done in a sequential way.

Step1

If $\mathsf{Verify}(vk, c_1||c_2||c_3||\mathsf{ID}||T||, \sigma) = \mathtt{reject}$, then $\mathcal{B}$ returns $\perp$ and skips **step2∼4**.

Step2

If $vk = vk^*$, then $\mathcal{B}$ stops and outputs random bit b'.

Step3

If $c_3 = c_3^*$, then $\mathcal{B}$ returns $\perp$.

Step4

$\mathcal{B}$ responds in the same way as in **Phase1**.

Guess If $\mathcal{A}$ outputs a bit, then $\mathcal{B}$ outputs the same bit as its guess.

We consider the $\mathcal{B}$'s simulation of the response to decryption queries in **Phase2**. In the case of Verify $= \mathtt{reject}$ in **Step1**, $\mathcal{B}$ returns $\perp$ in the same way as in our Decrypt algorithm, and then it perfectly simulates the challenger in Mal.KGC game. In the case of $vk = vk^*$ in **Step2**, the event $\mathtt{Forge}$ occurs. In the case of $c_3 = c_3^*$ in **Step3**, since c_3 equals to c_3^*, the decryption of c_3 is $M_0 = [(s_1 \oplus s_2 \oplus m_0)||vk^*]$ or $M_1 = [(s_1 \oplus s_2 \oplus m_1)||vk^*]$. However, since $vk \neq vk^*$, the decryption of c is $\perp$, and then $\mathcal{B}$ simulates perfectly. In the case of $c_3 \neq c_3^*$, $\mathcal{B}$ can issue the valid decryption query c_3 to $\mathcal{C}$.

If the event $\mathtt{Forge}$ does not occurs, $\mathcal{B}$ perfectly simulates the challengers in the $\mathsf{IND\text{-}ID\text{-}CCA_{TS}}$ game. Let $\mathtt{Succ}^{\mathcal{PKE}}$ denote the event that $\mathcal{B}$ wins the IND-CCA game.

We see that

$$\begin{aligned} Adv_{\Delta,\mathcal{B}}^{\mathsf{IND\text{-}CCA}} &= |\Pr[\mathtt{Succ}^{\mathcal{PKE}}] - \frac{1}{2}| \\ &\geq |\frac{1}{2} \cdot (1 - \frac{1}{q}) \\ &\quad + (\Pr[\mathtt{Succ} \wedge \overline{\mathtt{Forge}} + \frac{1}{2}\Pr[\mathtt{Forge}]) \cdot \frac{1}{q} - \frac{1}{2}| \end{aligned}$$

$$= \quad |\Pr[\mathtt{Succ} \wedge \overline{\mathtt{Forge}} + \frac{1}{2}\Pr[\mathtt{Forge}] - \frac{1}{2}| \cdot \frac{1}{q}.$$

$Adv^{\mathsf{IND\text{-}CCA}}_{\Delta,\mathcal{B}}$ is negligible since we assume Δ is IND-CCA secure. Therefore, $|\Pr[\mathtt{Succ} \wedge \overline{\mathtt{Forge}} + \frac{1}{2}\Pr[\mathtt{Forge}] - \frac{1}{2}|$ is also negligible. □

We see that

$$\begin{aligned} Adv^{\mathsf{Mal.KGC}}_{\Gamma,\mathcal{A}} &= |\Pr[\mathtt{Succ}] - \frac{1}{2}| \\ &= |\Pr[\mathtt{Succ} \wedge \mathtt{Forge}] - \frac{1}{2}\Pr[\mathtt{Forge}] \\ &\quad + \frac{1}{2}\Pr[\mathtt{Forge}] + \Pr[\mathtt{Succ} \wedge \overline{\mathtt{Forge}}] - \frac{1}{2}| \\ &\leq |\Pr[\mathtt{Succ} \wedge \mathtt{Forge}] - \frac{1}{2}\Pr[\mathtt{Forge}]| \\ &\quad + |\Pr[\mathtt{Succ} \wedge \overline{\mathtt{Forge}}] + \frac{1}{2}\Pr[\mathtt{Forge}] - \frac{1}{2}| \\ &\leq \frac{1}{2}\Pr[\mathtt{Forge}] \\ &\quad + |\Pr[\mathtt{Succ} \wedge \overline{\mathtt{Forge}}] + \frac{1}{2}\Pr[\mathtt{Forge}] - \frac{1}{2}|. \end{aligned}$$

$Adv^{Mal.KGC}_{\Gamma,\mathcal{A}}$ is negligible from **Claim 1** and **Claim 2**. This completes the proof of **Theorem 1**. □

2) Mal.Receiver *security:*

Theorem 2: If Π' is an IND-ID-CCA secure identity-based encryption scheme and Σ is an OT-sEUF-CMA secure one-time signature scheme, then Γ is a Mal.Receiver secure timed-release certificateless encryption scheme.

The proof is almost the same as that of $\mathsf{IND\text{-}ID\text{-}CCA_{CR}}$ security in [3].

3) Outsider *security:*

Theorem 3: If Π is an IND-ID-CCA secure identity-based encryption scheme and Σ is an OT-sEUF-CMA secure one-time signature scheme, then Γ is a Outsider secure timed-release certificateless encryption scheme.

The proof is almost the same as that of $\mathsf{IND\text{-}ID\text{-}CCA_{CR}}$ security in [3].

VIII. Conclusion

In this paper, we introduced a notion of TRCLE and defined Mal.KGC security, Mal.Receiver security and Outsider security. Moreover, we showed a generic construction of TRCLE in which a constructed scheme achieves those security if the primitive PKE scheme is IND-CCA secure, the primitive IBE schemes are IND-ID-CCA secure and the primitive one-time signature scheme is OT-sEUF-CMA secure.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 26330160.

References

[1] T. May, "Timed-release crypto," Manuscript, February 1993.

[2] A. C.-F. Chan and I. F. Blake, "Scalable, server-passive, user-anonymous timed release cryptography," in *ICDCS 2005*. IEEE Computer Society, 2005, pp. 504–513.

[3] T. Oshikiri and T. Saito, "Timed-release identity-based encryption," *IPSJ Journal*, vol. 55, no. 9, pp. 1964–1970, sep 2014.

[4] ——, "Timed-release hierarchical identity-based encryption," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 5, no. 11, pp. 148–154, 2014.

[5] Y. Dodis and J. Katz, "Chosen-ciphertext security of multiple encryption," in *TCC 2005*, ser. Lecture Notes in Computer Science, J. Kilian, Ed., vol. 3378. Springer-Verlag, 2005, pp. 188–209.

[6] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *ASIACRYPT 2003*, ser. Lecture Notes in Computer Science, C. S. Laih, Ed., vol. 2894. Springer-Verlag, 2003, pp. 452–473.

[7] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *STOC '90*. ACM, 1990, pp. 427–437.

[8] A. Shamir, "Identity-based cryptosystems and signature schemes," in *CRYPTO '84*, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds., vol. 196. Springer-Verlag, 1985, pp. 47–53.

[9] D. Boneh and M. K. Franklin, "Identity-based encryption from the Weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 584–615, 2003, a preliminary version appeared in *CRYPTO 2001*, 2001.

[10] R. C. Merkle, "A digital signature based on a conventional encryption function," in *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, ser. CRYPTO '87. London, UK, UK: Springer-Verlag, 1988, pp. 369–378. [Online]. Available: http://dl.acm.org/citation.cfm?id=646752.704751