Timed-Release Certificateless 暗号

押切 徹 †　　齊藤 泰一 †

† 東京電機大学
120-8551　東京都足立区千住旭町 5 番
13kmc06@ms.dendai.ac.jp

あらまし Timed-Release 暗号 (Timed-Release Encryption, TRE) とは，暗号化の際に復号できる時刻を指定できる暗号方式である．本稿では Certificateless 暗号に TRE の機能を持たせた Timed-Release Certificateless 暗号を提案し，その安全性定義と一般的構成方法を示す．

Timed-Release Certificateless Encryption

Toru Oshikiri†　　Taiichi Saito†

†Tokyo Denki University.
Senju Asahi-cho 5, Adachi-ku, Tokyo, 120–8551 JAPAN
13kmc06@ms.dendai.ac.jp

Abstract Timed-Release Encrtption(TRE) is an encryption mechanism that allows a reciever to decrypt a ciphertext only after the time that a sender designates. We propose a notion of certificateless encryption scheme with TRE encryption mechanism, Timed-Release Certificateless Encryption(TRCLE), and define its security model. Moreover we show a secure generic construction of TRCLE.

1　はじめに

Timed-Release 暗号 (Timed-Release Encryption, TRE) は暗号文を復号できる時刻を暗号化の際に指定できる暗号方式である．応用としては, ネットワークを利用した新作コンテンツの配信，オンライン試験における問題配布など，受信者の復号時刻が重要な意味を持つものがある．

TRE に関しては様々な研究が行われており，そのほとんどは，公開鍵暗号 (Public-Key Encryption, PKE) に TRE の機能を持たせた公開鍵型 Timed-Release 暗号 (Timed-Release Public-Key Encryption, TRPKE) に関するものである．しかし TRPKE は，PKE と同じく，暗号化するために受信者の公開鍵を事前に入手する必要がある．そのため，認証局 (Certificate Authority, CA) 等の公開鍵の正当性を保証する機関が必要である．

そこで我々は ID ベース暗号 (Identity-Based Encryption, IBE) に TRE の機能を持たせた，ID ベース型 Timed-Release 暗号 (Timed-Release Identity-Based Encryption, TRIBE) を提案した [3]．TRIBE では IBE と同じく公開鍵として受信者の識別子 ID(メールアドレスなど) を用いるため，公開鍵の事前入手・認証が不要である．しかし，時刻鍵がブロードキャストされた後は，KGC は全ユーザの暗号文を復号することが可能であるという安全性上の問題があった．

そこで本稿では Certificateless 暗号 (Certificateless Encryption, CLE) に TRE の機能を持たせた，Timed-Release Certificateless 暗号 (Timed-Release Certificateless Encryption, TR-

CLE) を提案する．TRCLE は，CLE と同じく CA が不要であり，KGC は部分秘密鍵と呼ばれる秘密鍵の一部分のみを生成し，最終的な秘密鍵はユーザ自身が生成する．

TRCLE は鍵生成センター (Key Generation Center, KGC)，時刻サーバ (Time-Server, TS)，送信者，受信者から構成される．

2 関連研究

[3] において我々は TRIBE を提案した．TRIBE の安全性については，時刻鍵が無ければ複号できないという受信者に対する IND-ID-CCA$_{\mathsf{CR}}$ 安全性，ユーザ秘密鍵が無ければ複号できないという TS に対する IND-ID-CCA$_{\mathsf{TS}}$ 安全性を定義し，さらにこの安全性を満たす TRIBE の一般的構成方法を示した．

今回提案する TRCLE の構成法は Dodis-Katz [2] の Multiple Encryption を利用した Dent [1] の CLE の構成法を参考にしている．

3 準備

本節では提案方式の構成に必要な公開鍵暗号，ID ベース暗号，One-Time 署名とその安全性について説明する．

3.1 公開鍵暗号

公開鍵暗号 Λ は，以下のアルゴリズム (PKE.Gen, PKE.Enc，PKE.Dec) で構成される．

PKE.Gen(1^k)： セキュリティパラメータ 1^k を入力とし，公開鍵 pk と秘密鍵 sk を出力する．

PKE.Enc(pk, m)： 公開鍵 pk，メッセージ m を入力とし，暗号文 c を出力する．

PKE.Dec(sk, c)： 秘密鍵 sk，暗号文 c を入力とし，メッセージ m もしくは $\perp$ を出力する．

3.1.1 IND-CCA 安全性

公開鍵暗号 Λ の選択暗号文攻撃に対する識別不可能性 (Indistinguishability against chosen ciphertext attacks, IND-CCA) は，以下のチャレンジャー $\mathcal{C}$ と攻撃者 $\mathcal{A}$ のゲームを用いて定義される．

Setup $\mathcal{C}$ は $(pk, sk) \leftarrow \mathsf{PKE.Gen}(1^k)$ を実行し，$\mathcal{A}$ に pk を与え，sk を保持しておく．

Phase1 $\mathcal{A}$ は Decrypt クエリ c を $\mathcal{C}$ に出すことができる．$\mathcal{C}$ は $\mathsf{PKE.Dec}(sk, c)$ を計算し，その出力 m または $\perp$ を $\mathcal{A}$ に返す．

Challenge $\mathcal{A}$ は同じ長さのメッセージ m_0, m_1 を $\mathcal{C}$ に送る．$\mathcal{C}$ は $b \in \{0, 1\}$ をランダムに選び，c^*=$\mathsf{PKE.Enc}(pk, m_b)$ を計算し，c^* を $\mathcal{A}$ に返す．

Phase2 $\mathcal{A}$ は Phase1 と同様に，Decrypt クエリを出すことができる．ただし，チャレンジ暗号文である c^* を出すことはできない．

Guess $\mathcal{A}$ は推測値 $\tilde{b}$ を出力する．

もし，$b = \tilde{b}$ ならば $\mathcal{A}$ の勝ちとする．
$\mathcal{A}$ のアドバンテージを以下の様に定義する．

$$Adv^{IND\text{-}CCA}_{\Lambda,\mathcal{A}}(1^k) = 2\Pr[b = \tilde{b}] - 1$$

定義 1 公開鍵暗号 Λ に対する攻撃者 $\mathcal{A}$ のアドバンテージ $Adv^{IND\text{-}CCA}_{\Lambda,\mathcal{A}}(1^k)$ が *negligible* であるとき，PKE は IND-CCA 安全であるという．

3.2 ID ベース暗号

ID ベース暗号 Π は，以下のアルゴリズム (IBE.Setup，IBE.Ext，IBE.Enc，IBE.Dec) で構成される．

IBE.Setup(1^k)： セキュリティパラメータ 1^k を入力とし，公開パラメータ $params$ とマスター秘密鍵 msk を出力する．

IBE.Ext($params, msk, id$)： 公開パラメータ $params$，マスター秘密鍵 msk ，id を入力とし，id に対応するユーザ秘密鍵 d_{id} を出力する．

IBE.Enc($params$, id, m)： 公開パラメータ $params$，id，メッセージ m を入力とし，暗号文 c を出力する．

IBE.Dec($params$, d_{id}, c)： 公開パラメータ $params$，ユーザ秘密鍵 d_{id}，暗号文 c を入力とし，メッセージ m もしくは $\perp$ を出力する．

3.2.1 IND-ID-CCA 安全性

ID ベース暗号 Π の選択暗号文攻撃および適応的 ID 攻撃に対する識別不可能性 (Indistinguishability against identity and chosen- ciphertext attacks, IND-ID-CCA) は，以下のチャレンジャー $\mathcal{C}$ と攻撃者 $\mathcal{A}$ のゲームを用いて定義される．

Setup $\mathcal{C}$ は $(params, msk) \leftarrow \mathsf{IBE.Setup}(1^k)$ を実行し，$\mathcal{A}$ に $params$ を与え，msk を保持しておく．

Phase1 $\mathcal{A}$ は Extract クエリ id と Decrypt クエリ (id, c) を $\mathcal{C}$ に出すことができる．Extract クエリ id に対し，$\mathcal{C}$ は ユーザ秘密鍵 $d_{id} = \mathsf{IBE.Ext}(params, msk, id)$ を計算し，d_{id} を $\mathcal{A}$ に返す．Decrypt クエリ (id, c) に対し，$\mathcal{C}$ は $d_{id} = \mathsf{IBE.Ext}(params, msk, id)$ を計算した後，$\mathsf{IBE.Dec}(d_{id}, c)$ を計算し，その出力 m または $\perp$ を $\mathcal{A}$ に返す．

Challenge $\mathcal{A}$ は同じ長さのメッセージ m_0, m_1 と Extract クエリを出していない id^* を $\mathcal{C}$ に送る．$\mathcal{C}$ は $b \in \{0, 1\}$ をランダムに選び，c^*=$\mathsf{IBE.Enc}(params, id^*, m_b)$ を計算し，c^* を $\mathcal{A}$ に返す．

Phase2 $\mathcal{A}$ は Phase1 と同様に，Extract クエリと Decrypt クエリを出すことができる．ただし，Extract クエリとしてチャレンジ ID である id^* を，Decrypt クエリとして **Challenge** で現れた (id^*, c^*) を，出すことはできない．

Guess $\mathcal{A}$ は推測値 $\tilde{b}$ を出力する．

もし，$b = \tilde{b}$ ならば $\mathcal{A}$ の勝ちとする．
$\mathcal{A}$ のアドバンテージを以下の様に定義する．

$$Adv_{\Pi,\mathcal{A}}^{IND\text{-}ID\text{-}CCA}(1^k) = 2\Pr[b = \tilde{b}] - 1$$

定義 2 *ID* ベース暗号 Π に対する攻撃者 $\mathcal{A}$ のアドバンテージ $Adv_{\Pi,\mathcal{A}}^{IND\text{-}ID\text{-}CCA}(1^k)$ が *negligible* であるとき，*IBE* は IND-ID-CCA 安全であるという．

3.3 One-Time 署名

One-Time 署名 Σ は，以下のアルゴリズム (SigGen，Sign，Verify) で構成される．

SigGen(1^k)： セキュリティパラメータ 1^k を入力とし，検証鍵 vk と署名鍵 sk を出力する．

Sign(sk, m)： 署名鍵 sk，メッセージ m を入力とし，署名 σ を出力する．

Verify(vk, m, σ)： 検証鍵 vk，メッセージ m，署名 σ を入力とし，`accept`/`reject` を出力する．

3.3.1 OT-sEUF-CMA 安全性

One-Time 署名 Σ の選択文書攻撃に対する強存在的偽造困難性 (One-time strong existential unforgeability against chosen message attacks, OT-sEUF-CMA) は，以下のチャレンジャー $\mathcal{C}$ と偽造者 $\mathcal{F}$ のゲームを用いて定義される．

Setup $\mathcal{C}$ は $(vk, sk) \leftarrow \mathsf{SigGen}(1^k)$ を実行し，$\mathcal{F}$ に vk を与え，sk を保持しておく．

Query $\mathcal{F}$ は $\mathcal{C}$ に対して 1 回だけメッセージ m に対する署名クエリを出すことができる．$\mathcal{C}$ は署名 σ=$\mathsf{Sign}(sk, m)$ を計算し σ を $\mathcal{A}$ に返す．

Forge $\mathcal{F}$ はメッセージと署名の組 (m^*, σ^*) を出力する．

$\mathcal{F}$ のアドバンテージを以下の様に定義する．
$Adv_{\Sigma,\mathcal{F}}^{OT\text{-}sEUF\text{-}CMA}(1^k) = \Pr[\mathsf{Verify}(vk, m^*, \sigma^*) = \mathtt{accept} \wedge (m, \sigma) \neq (m^*, \sigma^*)]$

定義 3 *One-Time* 署名 Σ に対する攻撃者 $\mathcal{F}$ のアドバンテージ $Adv_{\Sigma,\mathcal{F}}^{OT\text{-}sEUF\text{-}CMA}(1^k)$ が *negligible* であるとき，*One-Time* 署名は OT-sEUF-CMA 安全であるという．

4 提案

本節では Timed-Release Certificateless 暗号とその安全性定義を記述する．

4.1 Timed-Release Certificateless 暗号 (TRCLE)

Timed-Release Certificateless 暗号 Γ は，以下の 9 つのアルゴリズム (KGC.Setup, TS.Setup, Extract-Partial-Private-Key, Time-Signal-Release, Set-Secret-Value, Set-Public-Key, Set-Private-Key, Encrypt, Decrypt) で構成される．

KGC.Setup(1^k) : セキュリティパラメータ 1^k を入力とし，公開パラメータ $params$ とマスター秘密鍵 msk を出力する．

TS.Setup(1^k) : セキュリティパラメータ 1^k を入力とし，TS の公開鍵 tpk と秘密鍵 tsk を出力する．

Extract-Partial-Private-Key($params$, msk, id) : 公開パラメータ $params$，マスター秘密鍵 msk，id を入力とし，id に対応する部分秘密鍵 d_{id} を出力する．

Time-Signal-Release(tpk, tsk, t) : TS の公開鍵 tpk と秘密鍵 tsk，時刻 t を入力とし，t に対応する時刻鍵 d_t を出力する．

Set-Secret-Value($params$, id) : 公開パラメータ $params$ と id を入力とし，id に対応する秘密情報 x_{id} を出力する．

Set-Public-Key($params$, x_{id}) : 公開パラメータ $params$ と秘密情報 x_{id} を入力とし，ユーザ公開鍵 pub_{id} を出力する．

Set-Private-Key($params$, d_{id}, x_{id}) : 公開パラメータ $params$，部分秘密鍵 d_{id}，秘密情報 x_{id} を入力とし，ユーザ秘密鍵 prv_{id} を出力する．

Encrypt($params$, tpk, id, pub_{id}, t, m) : 公開パラメータ $params$，タイムサーバの公開鍵 tpk，id，ユーザ公開鍵 pub_{id}，指定時刻 t，メッセージ m を入力とし，暗号文 c を出力する．

Decrypt($params$, tpk, c, prv_{id}, d_t) : 公開パラメータ $params$，タイムサーバの公開鍵 tpk，暗号文 c，ユーザ秘密鍵 prv_{id}，時刻鍵 d_t を入力とし，メッセージ m もしくは $\perp$ を出力する．

4.2 TRCLE の安全性

TRCLE は部分秘密鍵 d_{id} と秘密情報 x_{id}，時刻鍵 d_t の 3 つの鍵が揃った時にのみ，復号できる暗号方式である．また TRCLE では CA が存在しないため，公開鍵のすり替えが起こり得る．そのため，以下の 4 つのエンティティに対する安全性を考慮する必要がある．

- KGC に対する安全性

  任意の部分秘密鍵を利用できても，id に対応する秘密情報無しでは暗号文からメッセージの情報を得ることができない

- TS に対する安全性

  任意の時刻鍵を利用できても，部分秘密鍵と id に対応する秘密情報無しでは暗号文からメッセージの情報を得ることができない

- 受信者に対する安全性

  正当なユーザ秘密鍵を持っていても，時刻鍵無しでは暗号文からメッセージの情報を得ることができない

- 外部者に対する安全性

  公開鍵のすり替えを行っても TS からブロードキャストされる時刻鍵だけでは暗号文からメッセージの情報を得ることができない

本稿では紙面の都合上，悪意のある KGC（Malicious KGC）に対する安全性 (Mal.KGC 安全性) のみ記述する．

4.2.1 Mal.KGC 安全性

Timed-Release Certificateless 暗号 Γ の KGC に対する Mal.KGC 安全性は，以下のチャレンジャー $\mathcal{C}$ と攻撃者 $\mathcal{A}$ のゲームを用いて定義される．

Setup $\mathcal{C}$ は $(params, msk) \leftarrow \mathsf{KGC.Setup}(1^k)$, $(tpk, tsk) \leftarrow \mathsf{TS.Setup}(1^k)$ を実行し，$\mathcal{A}$ に $params$, msk, tpk を与え，tsk を保持しておく．

Phase1 $\mathcal{A}$ は以下の 4 つのクエリを行うことができる．

Request Public Key クエリ $\mathcal{A}$ は id を $\mathcal{C}$ に送る．
$\mathcal{C}$ は $x_{id} \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(params, id)$, $pub_{id} \leftarrow \mathsf{Set\text{-}Public\text{-}Key}(params, x_{id})$ を計算し，pub_{id} を $\mathcal{A}$ に返す．

Time Signal Release クエリ $\mathcal{A}$ は t を $\mathcal{C}$ に送る．
$\mathcal{C}$ は $d_t \leftarrow \mathsf{Time\text{-}Signal\text{-}Release}(tpk, tsk, t)$ を計算し，d_t を $\mathcal{A}$ に返す．

Extract Secret Value クエリ $\mathcal{A}$ は id を $\mathcal{C}$ に送る．$\mathcal{C}$ は $x_{id} \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(params, id)$ を計算し，x_{id} を $\mathcal{A}$ に返す．

Decrypt クエリ $\mathcal{A}$ は (id, t, c) を $\mathcal{C}$ に送る．
$\mathcal{A}$ は
$d_{id} \leftarrow \mathsf{Extract\text{-}Partial\text{-}Private\text{-}Key}(params, msk, id)$,
$x_{id} \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(params, id)$,
$prv_{id} \leftarrow \mathsf{Set\text{-}Private\text{-}Key}(params, d_{id}, x_{id})$
$d_t \leftarrow \mathsf{Time\text{-}Signal\text{-}Release}(tpk, tsk, t)$
を計算した後，$\mathsf{Decrypt}(id, params, tpk, c, prv_{id}, d_t)$ を計算し，その出力 m または $\perp$ を $\mathcal{A}$ に返す．

Challenge $\mathcal{A}$ は同じ長さの任意メッセージ m_0, m_1 と id^*，指定時刻 t^* を $\mathcal{C}$ に送る．$\mathcal{C}$ は $b \in \{0,1\}$ をランダム選び，$c^* = \mathsf{Encrypt}(params, tpk, id^*, pub_{id^*}, t^*, m_b)$ を計算し，c^* を $\mathcal{A}$ に返す．

Phase2 $\mathcal{A}$ は **Phase1** と同様に，**Request Public Key クエリ**と **Time Signal Release クエリ**，**Extract Secret Value クエリ**，**Decrypt クエリ**を出すことができる．ただし，**Extract Secret Value クエリ**としてチャレンジ ID である id^* を，**Decrypt クエリ**として **Challenge** で現れた (id^*, t^*, c^*) を，出すことはできない．

Guess $\mathcal{A}$ は推測値 $\tilde{b}$ を出力する．

もし，$b = \tilde{b}$ ならば $\mathcal{A}$ の勝ちとする．
$\mathcal{A}$ のアドバンテージを以下の様に定義する．

$$Adv^{Mal.KGC}_{\Gamma,\mathcal{A}}(1^k) = 2\Pr[b = \tilde{b}] - 1$$

定義 4 *Timed-Release Certificateless* 暗号 Γ に対する攻撃者 $\mathcal{A}$ のアドバンテージ $Adv^{Mal.KGC}_{\Gamma,\mathcal{A}}(1^k)$ が *negligible* であるとき，*TRCLE* Γ は *Mal.KGC* 安全であるという．

5 一般的構成法

本節では公開鍵暗号 $\Lambda = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ と 2 つの ID ベース暗号 $\Pi = (\mathsf{IBE.Setup}, \mathsf{IBE.Ext}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$, $\Pi' = (\mathsf{IBE'.Setup}, \mathsf{IBE'.Ext}, \mathsf{IBE'.Enc}, \mathsf{IBE'.Dec})$ と One-Time 署名 $\Sigma = (\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$ を用いた Timed-Release Certificateless 暗号 Γ の構成法を示す．

5.1 構成

$\mathsf{KGC.Setup}(1^k)$**:**
Step 1: Run $\mathsf{IBE.Setup}$ on input 1^k to generate $(params, msk)$.
Step 2: Return $(params, msk)$.

$\mathsf{TS.Setup}(1^k)$**:**
Step 1: Run $\mathsf{IBE'.Setup}$ on input 1^k to generate $(params, msk)$.
Step 2: Set $tpk = params$ and $tsk = msk$.
Step 3: Return (tpk, tsk).

$\mathsf{Extract\text{-}Partial\text{-}Private\text{-}Key}(params, msk, id)$**:**
Step 1: Run $\mathsf{IBE.Ext}(params, msk, id)$ to obtain d_{id}.
Step 2: Return d_{id}.

Time-Signal-Release(tpk, tsk, t)**:**

Step 1: Run IBE′.Ext(tpk, tsk, t) to obtain d_t.

Step 2: Return d_t.

Set-Secret-Value($params, id$)**:**

Step 1: Run PKE.Gen on input 1^k to generate (pk_{id}, sk_{id}).

Step 2: Set $x_{id} = (pk_{id}, sk_{id})$.

Step 3: Return x_{id}.

Set-Public-Key($params, x_{id}$)**:**

Step 1: Parse $x_{id} = (pk_{id}, sk_{id})$.

Step 2: Set $pub_{id} = pk_{id}$.

Step 3: Return pub_{id}.

Set-Private-Key($params, d_{id}, x_{id}$)**:**

Step 1: Parse $x_{id} = (pk_{id}, sk_{id})$.

Step 2: Set $prv_{id} = (d_{id}, sk_{id})$.

Step 3: Return prv_{id}.

Encrypt($params, tpk, id, pub_{id}, t, m$)**:**

Step 1: Run SigGen on input 1^k 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($params$, $s_1||vk$, id).

Step 5: Compute $c_2 = \mathsf{IBE'.Enc}(tpk, s_2||vk, t)$.

Step 6: Compute $c_3 = \mathsf{PKE.Enc}(pub_{id}, s_3||vk)$.

Step 7: Compute $\sigma = \mathsf{Sign}(sk, c_1||c_2||c_3||id||t)$.

Step 8: Set $c = (c_1, c_2, c_3, id, t, vk, \sigma)$.

Step 9: Return c.

Decrypt($params, tpk, c, prv_{id}, d_t$)**:**

Step 1: If $\mathsf{Verify}(vk, c_1||c_2||c_3||id||t, \sigma) =$ `reject` , then return $\perp$ and stop.

Step 2: Compute $s_1||vk' = \mathsf{IBE.Dec}(params,$ d_{id}, $c_1)$.

Step 3: Compute $s_2||vk'' = \mathsf{IBE'.Dec}(tpk,$ d_t, $c_2)$.

Step 4: Compute $s_3||vk''' = \mathsf{PKE.Dec}(sk_{id},$ $c_3)$.

Step 5: If $vk = vk' = vk'' = vk'''$, then return $m = s_1 \oplus s_2 \oplus s_3$ else return $\perp$.

5.2 構成された TRCLE の安全性

5.2.1 Mal.KGC 安全性

定理 1 公開鍵暗号 Λ が IND-CCA 安全，*One-Time* 署名 Σ が OT-sEUF-CMA 安全ならば，提案方式 Γ は Mal.KGC 安全である．

証明 $\mathcal{A}$ を Γ の Mal.KGC 安全性に対する攻撃者とする．また暗号文 $c = (c_1, c_2, c_3, id, t, vk, \sigma)$ が $\mathsf{Verify}(vk, c_1||c_2||c_3||id||t, \sigma) =$ `accept` を満たす場合，この暗号文を正当な暗号文と呼ぶこととし，$c^* = (c_1^*, c_2^*, c_3^*, id^*, t^*, vk^*, \sigma^*)$ をチャレンジ暗号文とする．ここで 3 つのイベント `Forge`，`Succ`，`Guess` を定義する．

`Forge`：$\mathcal{A}$ が **Phase2** において復号オラクルに正当な暗号文 $(c_1, c_2, c_3, id, t, vk^*, \sigma)$ を問い合わせる．

`Succ`：$\mathcal{A}$ が Mal.KGC ゲームに勝利する．

`Guess`:**Setup** でランダムに選んだインデックス i に対して，i 番目の **Request Public Key** クエリの ID が **Challenge** ID に一致する．

また **Request Public Key** クエリの回数を q_{req} とする．このとき，以下の 2 つの補題が成立する．

補題 1 $\Pr[\mathtt{Forge}]$ は *negligible* である．

補題 2 $|\Pr[\mathtt{Succ} \wedge \overline{\mathtt{Forge}}|\mathtt{Guess}] - \frac{1}{2}| \cdot \frac{1}{q_{req}}$ は *negligible* である．

この 2 つの補題より，

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

が成立し，定理 1 の証明が完了する．

証明[補題 1]

イベント `Forge` が起こる仮定すると，Mal.KGC 攻撃者 $\mathcal{A}$ を使って OT-sEUF-CMA の安全性を破る署名偽造者 $\mathcal{F}$ を構成できることを示す．

Setup チャレンジャー $\mathcal{C}$ は
$(vk^*, sk^*) \leftarrow \mathsf{SigGen}(1^k)$ を実行し，$\mathcal{F}$ に vk^* を渡す．$\mathcal{F}$ は $\mathcal{A}$ に対して Mal.KGC ゲームをシミュレートするため
$(params, msk) \leftarrow \mathsf{KGC.Setup}(1^k)$,
$(tpk, tsk) \leftarrow \mathsf{TS.Setup}(1^k)$ を実行し，
$\mathcal{A}$ に $(params,\ msk,\ tpk)$ を与える．

Query $\mathcal{A}$ の **Request Public Key** クエリと **Time Signal Release** クエリ，**Extract Secret Value** クエリ，**Decrypt** クエリに対して，$\mathcal{F}$ は tsk と msk を持っているため全て答えることができる．

Challenge $\mathcal{A}$ がチャレンジとして $(m_0,\ m_1,\ id^*,\ t^*)$ を出力した場合，$\mathcal{F}$ は $b \in \{0,1\}$ と，メッセージと同じ長さの 2 つの値 $s_1 \in \{0,1\}^{|m|}$，$s_2 \in \{0,1\}^{|m|}$ をランダムに選ぶ．
$s_3 = s_1 \oplus s_2 \oplus m_b$ を計算し，
$c_1^* = \mathsf{IBE.Enc}(params, s_1||vk^*, id^*)$,
$c_2^* = \mathsf{IBE'.Enc}(tpk, s_2||vk^*, t^*)$,
$c_3^* = \mathsf{PKE.Enc}(pub_{id}, s_3||vk^*)$ を計算する．
$m^* = (c_1||c_2||c_3||id^*||t^*)$ を $\mathcal{F}$ の OT-sEUF-CMA ゲームの署名クエリとして出力し，m^* に対する署名 σ^* を得る．最後に $\mathcal{F}$ はチャレンジ暗号文として $c^* = (c_1^*,\ c_2^*,\ c_3^*,\ id^*,\ t^*,\ vk^*,\ \sigma^*)$ を $\mathcal{A}$ に返す．

Forge イベント `Forge` が起こるという仮定より，$\mathcal{A}$ は **Phase2** において正当な暗号文 $c = (c_1, c_2, c_3, id, t, vk^*, \sigma)$ を Decrypt オラクルに問い合わせる．この時，Mal.KGC ゲームの Decrypt オラクルの入力制限 $c \neq c^*$ より $(c_1, c_2, c_3, id, t, \sigma) \neq (c_1^*,\ c_2^*,\ c_3^*,\ id^*,\ t^*,\ \sigma^*)$ が成立していなければならない為，$\mathcal{F}$ は偽造署名として
$(c_1||c_2||c_3||id||t, \sigma)$ を出力する．

上記より `Forge` が起こると仮定すると，署名偽造者 $\mathcal{F}$ を構成できる．しかしこれは OT-sEUF-CMA 安全性に反する．よってイベント `Forge` が起こる確率は negligible である．

証明[補題 2]

$\mathcal{A}$ を Mal.KGC 安全性に対する攻撃者と仮定すると，この $\mathcal{A}$ を利用して Λ の IND-CCA 安全性を破る攻撃者 $\mathcal{B}$ を構成できることを示す．

証明の簡素化のため，**Request Public Key** クエリを行なっていない id を **Time Signal Release** クエリ，**Extract Secret Value** クエリ，**Decrypt** クエリ，**Challenge** に出すことはできないと仮定する．

Setup IND-CCA ゲームのチャレンジャー $\mathcal{C}$ は
$(pk^*, sk^*) \leftarrow \mathsf{PKE.Gen}(1^k)$ を実行し，$\mathcal{B}$ に pk^* を与える．$\mathcal{B}$ は $\mathcal{A}$ に対して Mal.KGC ゲームをシミュレーションするため
$(params, msk) \leftarrow \mathsf{KGC.Setup}(1^k)$,
$(tpk,\ tsk) \leftarrow \mathsf{TS.Setup}(1^k)$ を実行し，$\mathcal{A}$ に $(params, msk, tpk)$ を与える．また $\mathcal{B}$ はインデックス $i \in \{1, 2, \cdots q_{req}\}$ をランダムに選ぶ．

Phase1 $\mathcal{A}$ の 4 つのクエリに対して，$\mathcal{B}$ は以下のように振舞う．

Request Public Key クエリ i 番目のクエリに対しては，IND-CCA ゲームのチャレンジ公開鍵 pk^* を $\mathcal{A}$ に返す．
i 番目以外のクエリに対しては
$(pk_{id}, sk_{id}) \leftarrow \mathsf{PKE.Gen}(1^k)$ を実行し，pk_{id} を $\mathcal{A}$ に返す．生成した (pk_{id}, sk_{id}) を保管しておく．

Time Signal Release クエリ
$d_t \leftarrow \mathsf{IBE'.Ext}(tpk, tsk, t)$ を実行し，d_t を $\mathcal{A}$ に返す．

Extract Secret Value クエリ **Request Public Key クエリ**の際に生成した (pk_{id}, sk_{id}) を x_{id} として $\mathcal{A}$ に返す．

Decrypt クエリ
$\mathsf{Verify}(vk, c_1||c_2||c_3||id||t, \sigma) = \texttt{reject}$ ならば $\perp$ を返す．そうでなければ，
$s_1||vk' \leftarrow \mathsf{IBE.Dec}(c_1, d_{id})$,
$s_2||vk'' \leftarrow \mathsf{IBE.Dec}(c_2, st)$ を計算する．ここで id が i 番目であった場合は，c_3 を $\mathcal{B}$ の **Decrypt** クエリとして出力し，$s_3||vk'''$ を受け取る．i 番目以外の場合は，$s_3||vk''' \leftarrow \mathsf{PKE.Dec}(c_3,\ sk_{id})$ を計算する．$vk = vk' = vk'' = vk'''$ ならば，$m = s_1 \oplus s_2 \oplus s_3$ を計算し，m を $\mathcal{A}$ に返す．そうでなければ $\perp$ を返す．

Challenge $\mathcal{A}$ が Challenge として (m_0, m_1, id^*, t^*) を出力する．id^* が i 番目以外の id であった場合は，シミュレートを停止しランダムビット $\tilde{b}$ を出力する．そうでなければ $\mathcal{B}$ は $(sk^*, vk^*) \leftarrow \mathsf{SigGen}(1^k)$ を実行する．メッセージと同じ長さの値 $r_1 \in \{0,1\}^{|m|}$ と $r_2 \in \{0,1\}^{|m|}$ をランダムに選び，$s_1 = r_1||vk^*$，$s_2 = r_2||vk^*$，$M_0 = r_1 \oplus r_2 \oplus m_0||vk^*$, $M_1 = r_1 \oplus r_2 \oplus m_1||vk^*$ とする．$c_1^* = \mathsf{IBE.Enc}(params, s_1, id^*)$, $c_2^* = \mathsf{IBE'.Enc}(tpk, s_2, t^*)$ を計算する．(M_0, M_1) を $\mathcal{B}$ の **Challenge** として出力し，暗号文 c_3^* を受け取る．$\mathcal{B}$ は $\sigma^* = \mathsf{Sign}(sk^*, c_1^*||c_2^*||c_3^*||id^*||t^*)$ を計算し，$c^* = (c_1^*, c_2^*, c_3^*, id^*, t^*, vk^*, \sigma^*)$ をチャレンジ暗号文として $\mathcal{A}$ に返す．

Phase2 $\mathcal{A}$ の **Request Public Key** クエリと **Time Signal Release** クエリ，**Extract Secret Value** クエリに対しては **Phase1** と同様に動作する．**Decrypt** クエリに対しては，以下の Step1〜4 を順に実行する．
Step1. $\mathsf{Verify}(vk, c_1||c_2||c_3||id||t, \sigma) = \mathtt{reject}$ ならば $\perp$ を返す．
Step2. $vk = vk^*$ ならばシミュレートを停止しランダムビットを出力する．
Step3. $c_3 = c_3^*$ ならば $\perp$ を返す．
Step4. **Phase1** と同様に動作する．

Guess $\mathcal{B}$ は $\mathcal{A}$ が出力する推測値 $\tilde{b}$ をそのまま出力する．

上記の IND-CCA 攻撃者 $\mathcal{B}$ の構成における，**Phase2** での Mal.KGC 攻撃者 $\mathcal{A}$ の Decrypt クエリに対する $\mathcal{B}$ の振舞いについて説明する．
Step1 の場合：本構成の Decrypt アルゴリズムの Step1 の判定条件を満たさない．よって正しくシミュレートできている．
Step2 の場合：イベント Forge である．
Step3 の場合：$c_3 (= c_3^*)$ の復号結果は M_0 あるいは M_1 であるが，$vk \neq vk^*$ であるため，本構成の Decrypt アルゴリズムの Step5 の判定条件を満たさない．よって正しくシミュレートできている．
Step4 の場合：$c_3 \neq c_3^*$ が成立しているため，$\mathcal{B}$ の IND-CCA ゲームの **Decrypt** クエリとして c_3 を出すことができる．よって正しくシミュレートできている．

よって Forge が起こらない限り，完全なシミュレーションになっている．**Challenge** において id^* が i 番目でなかった場合はランダムビットを出力しているだけなので，$\mathtt{Succ}^{\mathcal{PKE}}$ を $\mathcal{B}$ が Λ の IND-CCA 安全性を破るイベントとすれば Forge が起こらない場合，

$$|\Pr[\mathtt{Succ}^{\mathcal{PKE}}] - \frac{1}{2}| = |\Pr[\mathtt{Succ} \wedge \overline{\mathtt{Forge}}|\mathtt{Guess}] - \frac{1}{2}| \cdot \frac{1}{q_{req}}$$

が成立する．以上より，$Adv_{\Gamma,\mathcal{A}}^{Mal.KGC}$ を無視できないと仮定すると，$\mathcal{B}$ が無視できない確率で Λ の IND-CCA 安全性を破ることがわかる．

6 まとめ

本稿では Timed-Release Certificateless 暗号の安全性を定義し，その構成方法を示した．その構成法は，公開鍵暗号と ID ベース暗号，One-Time 署名を用いた generic construction であり，構成された TRCLE 方式の安全性はスタンダードモデルで証明可能である．

参考文献

[1] Dent, A. W.: A survey of certificateless encryption schemes and security models, *International Journal of Information Security (IJIS)*, Vol. 7, No. 5, pp. 349–377 (2008).

[2] Dodis, Y. and Katz, J.: Chosen-Ciphertext Security of Multiple Encryption, *TCC 2005* (Kilian, J., ed.), Lecture Notes in Computer Science, Vol. 3378, Springer-Verlag, pp. 188–209 (2005).

[3] Oshikiri, T. and Saito, T.: Timed-Release Identity-Based Encryption, *IEICE Tech. Rep.*, ISEC2013-2, Vol. 113, No. 53, Tokyo, pp. 9–13 (2013).