Vol.2014-DPS-158 No.11
Vol.2014-CSEC-64 No.11
2014/3/6

Hierarchical Time-Released Proxy Re-Encryption

小澤 賢佑[1,a)]　齊藤 泰一[1,b)]

概要：復号時間を指定可能なマルチキャスト暗号通信を実現可能な公開鍵暗号方式として Timed-Release Proxy Re-Encryption(TR-PRE) がある。しかし、この方式は一つの Proxy に再暗号化処理が集中するという問題がある。 本稿では、複数の Proxy を階層的に配置することにより再暗号化処理を分散可能な公開鍵暗号方式” hierarchical Timed-Release Proxy Re-Encryption(hierarchicalTR-PRE)”を提案する。

1. はじめに

公開前の新作映画、企業の決算・株価情報、電子投票の集計結果などは、特定時刻まで情報を秘匿し、特定時刻後一斉に公開する必要がある。多くの場合このような情報は複数のユーザに対し公開されることが予想される。特定時刻まで秘密を守るための方法として考えられるのは、特定時刻までは物理的に秘密にしておき、特定時刻になると同時にネットなどの各媒体を通じ情報を公開する方法である。しかし最新の情報は公開直後に多数のユーザが一斉に求めることが多く、この方法ではシステムに多大な負荷が掛かってしまうことが予想される。

本研究は、暗号化の際に復号時刻を指定でき、多数のユーザに対し暗号化データを送信できる、復号時刻制御可能なマルチキャスト暗号方式 hierarchical Time-Released Proxy Re-Encryption(hTR-PRE) の実現を目指す。

復号時刻の制御とマルチキャスト暗号通信の両方を実現可能な公開鍵暗号方式として Timed-Release Proxy Re-Encryption(TR-PRE)[5] がある。しかし、この方式は一つの Proxy に再暗号化処理が集中するという問題がある。そこで本研究では multi-hop という性質を持つ Proxy Re-Encryption(PRE) を用いる。multi-hopPRE とは、再暗号化の処理を複数回行うことができる PRE である。例えばユーザ A 宛ての暗号文をユーザ B 宛てに、それをさらに別のユーザ C 宛てに変更することができる。この multi-hop PRE を用いて Proxy を階層的に配置することにより、負荷を複数の Proxy に分散し問題の解決を目指す。

[1] 東京電機大学,
Tokyou Denki Uniersity,Adachi,Tokyo,120-8551,Japan
[a)] 12kmc10@ms.dendai.ac.jp
[b)] taiichi@c.dendai.ac.jp

2. 関連技術

2.1 ID-based Encryption

ID-based Encryption(IBE) とは ID に基づく暗号方式 [3] のことである。ここで ID とは Identity または Identification を取り、個人を特定することが可能な情報を意味する。例えば氏名、email アドレス、電話の番号といったものは ID として扱うことが可能である。IBE は平文、受信者の ID 、共通パラメータのみを用いて受信者への暗号文を作成することができるので、公開鍵暗号のように公開鍵認証センタは必要としない。つまり公開鍵認証センタが公開鍵に与える信頼性のかわりに ID の信頼性を利用する方式である。IBE は (Setup,Extract,Enc,Dec) の 4 つのアルゴリズムで構成される。

- **Setup** ： $\mathsf{Setup}(1^k) \rightarrow (msk, params)$
  セキュリティパラメータ 1^k を入力とし，マスター秘密鍵と共通パラメータのペア $(msk, params)$ を出力する.
- **Extract** ：$\mathsf{Extract}(msk, ID, params) \rightarrow d_{ID}$
  マスター鍵 msk, ID と共通パラメータ $params$ を入力とし、ID に対応した復号鍵 d_{ID} を出力する。
- **Encryption** ：$\mathsf{Enc}(m, ID, params) \rightarrow C$
  平文 $m \in \mathcal{M}$，ID と共通パラメータ $params$ を入力とし、暗号文 C を出力する。ただし $\mathcal{M}$ は平文空間である。
- **Decryption** ：$\mathsf{Dec}(C, d_{ID}) \rightarrow C$
  暗号文 C と復号鍵 d_{ID} を入力とし，平文 m、もしくはエラーシンボル $\perp$ を出力する。

2.2 Proxy Re-Encryption

Proxy Re-Encryption(PRE) とは本来の暗号文の受信者 A が、受け取った暗号文を信頼できる第 3 者である Proxy に送信し、Proxy が受信者 A が指定したユーザ宛ての暗号文に変更できる機能を持つ暗号方式である。また以下のような性質を持つ。

- single-hop/multi-hop：再暗号化した暗号文を更に別のユーザの秘密鍵で復号できる暗号文に再暗号化できる方式を multi-hop、再暗号化をした暗号文を更に再暗号化することができない方式 single-hop という。
- bidirectinal/unidirectinal ：ユーザ A とユーザ B 間の再暗号化鍵を用いてユーザ A 宛ての暗号文 C_A をユーザ B 宛ての暗号文 C_B に変換でき、またユーザ B 宛ての暗号文 C'_B をユーザ A 宛ての暗号文 C'_A に変換できる方式を bidirectinal、どちらかしかできない方式を unidirectinal という。

multi-hop bidirectinal PRE は (KeyGen,ReKeyGen,Enc,ReEnc,Dec) の 5 つのアルゴリズムで構成される。

- **KeyGeneration**：$\mathsf{KeyGen}(1^k) \rightarrow (pk, sk)$
  セキュリティパラメータ 1^k を入力とし公開鍵 pk、秘密鍵 sk を出力する。
- **ReEncryptionKeyGeneration**：
  $\mathsf{ReKeyGen}(sk_A, sk_B) \rightarrow rk_{AB}$
  2 つの秘密鍵 sk_A, sk_B を入力とし、再暗号化鍵 rk_{AB} を出力する。
- **Encryption** ： $\mathsf{Enc}(pk, m) \rightarrow C$
  公開鍵 pk、メッセージ m を入力とし暗号文 C を出力する。
- **ReEncryption** ： $\mathsf{ReEnc}(rk_{AB}, C_A) \rightarrow C_B$
  再暗号化鍵 rk_{AB}、ユーザ A 宛の暗号文 C_A を入力としユーザ B 宛の暗号文 C_B を出力する。
- **Decryption** ： $\mathsf{Dec}(sk, C) \rightarrow m$
  秘密鍵 sk、暗号文 C を入力とし、メッセージ m を出力する。あるいは $\perp$ を出力する。

2.3 判定双線型 Diffie-Hellman（DBDH）仮定

DBDH 仮定とは DBDH 問題が難しいという仮定である。素数 p を位数としたの群 G, G_T にペアリング演算 $e(G \times G \rightarrow G_T)$ が定義されているとする。この時 $g \in G$ において $< g, g^a, g^b, g^c, T >$ の組が与えられたとき $T = e(g,g)^{abc}$ もしくは T がランダムである、と判定する問題を、判定双線型 Diffie-Hellman(DBDH) 問題という。DBDH 問題が解ける Algorithm $\mathcal{B}$ の成功確率 $Adv\mathcal{B}$ は次のようになる。$Adv\mathcal{B} = |Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1] - Pr[\mathcal{B}(g, g^a, g^b, g^c, R) = 1]|$. $AdvB$ が negligible であるとき DBDH 仮定が成り立つという。本研究では DBDH 問題を変更した Modified DBDH(mDBDH) 問題を用いる。具体的には $e(g,g)^{abc}$ としていたところを、$e(g,g)^{ab/c}$ とする。mDBDH 問題と DBDH 問題は等価である [1]。

3. 提案方式

3.1 定義

hTR-PRE は (Setup,KeyGen,TS-Release,ReKeyGen,Enc,ReEnc,Dec) の 7 つのアルゴリズムで構成される。

- **Setup**
  $\mathsf{Setup}(1^k) \rightarrow (params, ts_{priv})$
  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 $params$ とタイムサーバの秘密鍵 ts_{priv} を出力する。
- **KeyGeneration**
  $\mathsf{KeyGen}(1^k) \rightarrow (pk, sk)$
  セキュリティパラメータ 1^k を入力とし公開鍵 pk、秘密鍵 sk を出力する。
- **TimeServer-Release**
  $\mathsf{TS\text{-}Release}(params, ts_{priv}, T) \rightarrow (S_T)$
  $params, ts_{priv}$、開示時刻 T を入力とし、時刻鍵 S_T を出力する。
- **ReEncryptionKeyGeneration**
  $\mathsf{ReKeyGen}(params, sk_A, sk_B) \rightarrow (rk_{AB})$
  params、2 つの秘密鍵 sk_A, sk_B を入力とし、再暗号化鍵 rk_{AB} を出力する。
- **Encryption**
  $\mathsf{Enc}(params, pk, m, T) \rightarrow C$
  公開鍵 pk、メッセージ m、開示時刻 T を入力とし暗号文 C を出力する。
- **ReEncryption**
  $\mathsf{ReEnc}(params, rk_{AB}, C_A) \rightarrow C_B$
  params、再暗号化鍵 rk_{AB}、ユーザ A 宛の暗号文 C_A を入力としユーザ B 宛の暗号文 C_B を出力する。あるいは $\perp$ を出力する。
- **Decryption**
  $\mathsf{Dec}(params, sk, pk, C, S_T) \rightarrow m$
  params、秘密鍵 sk、公開鍵 pk、暗号文 C、時刻鍵 S_T を入力とし、メッセージ m を出力する。あるいは $\perp$ を出力する。

3.2 安全性定義

3.2.1 準備

入力条件を明確にするために公開鍵にラベルを付け、無向グラフ (V,E) を考える。Uncorrupted key generation オラクルへのクエリで得た pk にラベル“uncorrupted”を付ける。Corrupted key generation オラクルへのクエリで得た pk にラベル“corrupted”を付ける。チャレンジ公開鍵 pk^* にラベル“target”を付ける。次に無向グラフ (V,E) を定義する。

- $V = \{pk_A, pk_B, pk_C, ...\}$, $E \subseteq V \times V$

- V ：公開鍵の集合
- エッジ $(pk_A, pk_B) \in E \Leftrightarrow$“ユーザ A、ユーザ B 間で再暗号化鍵 rk_{AB} が生成済み”

以上のようなグラフの中で target を含む連結成分を“target グループ (Target)”、corrupted を含む連結成分を“Corrupted グループ (Corrupted)”、それ以外を“Uncorrupted グループ (Uncorrupted)”と呼ぶ。また、Target と Uncorrupted 間ではエッジが張られる場合があるがその場合連結成分すべてが Target となる。例えば、pk_A ∈Target と pk_B ∈Uncorupted 間で再暗号化鍵 rk_{AB} が生成された場合 pk_B を含むグループも Target となる。また暗号文 C′ を 1 回以上再暗号化を行い得た暗号文 C を“C⇐C′”のように書く。

3.2.2 Malicious TimeServer Security

hTR-PRE 方式 Γ(Setup,KeyGen,TS-Release,ReKeyGen, Enc,ReEnc,Dec) に対する Malicious TimeServer Security を定義する。多項式時間アルゴリズムである攻撃者 $\mathcal{A}$ は TimeServer の秘密鍵を所持しており、挑戦者 $\mathcal{CH}$ と以下のようなゲームを行う。

$\mathcal{A}$ には以下のオラクルへのアクセスを許す。

Uncorrupted key generatio

$\mathcal{A}$ がクエリすると $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ を実行し $\mathcal{A}$ に pk を返す。

Corrupted key generation

$\mathcal{A}$ がクエリすると $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ を実行し $\mathcal{A}$ に (pk, sk) を返す。

Re-Encryption key generation

(pk_A, pk_B) をオラクルへの入力とし、オラクルは再暗号化鍵 $rk_{AB} \leftarrow \mathsf{ReKeyGen}(sk_A, sk_B)$ を返す。ただし“pk_A ∈Uncorrupted かつ pk_B ∈Corrupted”または“pk_A ∈Corrupted かつ $pk_B \in$ Uncorrupted”の場合 $\perp$ を出力する。

Challenge

このオラクルへのクエリは 1 回のみ許される。$(pk^*, T^*, m_0^*, m_1^*)$ をオラクルへの入力とする。pk^* をチャレンジ key とする。オラクルは $b \leftarrow \{0, 1\}$ をランダムに決定しチャレンジ暗号文 $C^* \leftarrow \mathsf{Enc}(pk^*, m_b^*, T^*)$ を返す。

Re-Encryption

(pk_A, pk_B, C_A) をオラクルへの入力とし、ユーザ B 宛の暗号文 $C_B \leftarrow \mathsf{ReEnc}(\mathsf{ReKeyGen}(sk_A, sk_B), C_A)$ を返す。ただし、pk_A ∈Target, pk_B ∈Corrupted, $C_A \Leftarrow C^*$ がすべて成り立つ場合場合 $\perp$ を返す。

Decryption

(pk, T, C) をオラクルへの入力するとし、メッセージ $m = \mathsf{Dec}(sk, pk, C, S_T)$ を返す。ただし $C = C^*$ または $C \Leftarrow C^*$ の場合は⊥を出力する。

Decision

$pk*$ ∈Corrupted の場合は $\perp$、そうでないならば $\mathcal{A}$ は b' を予想し出力する。

もし b' と b が等しければ $\mathcal{A}$ の勝利となる。$\mathcal{A}$ の勝利確率を次のように定義する。$Adv_{\Gamma,\mathcal{A}}(1^k) = 2Pr[b = b'] - 1$

定義 1：hTR-PRE 方式 Γ に対する任意の多項式時間アルゴリズム $\mathcal{A}$ の勝利確率 $Adv_{\Gamma,\mathcal{A}}(1^k)$ が negligible である時 Γ は Malicious TimeServer Security を満たすという。

3.2.3 Malicious User Security

hTR-PRE 方式 Γ に対する Malicious User Security を定義する。多項式時間アルゴリズムである攻撃者 $\mathcal{A}$ はすべてのユーザの秘密鍵、公開鍵を得ることができ、挑戦者 $\mathcal{CH}$ と以下のようなゲームを行う。

Keygeneration

$\mathcal{A}$ がクエリすると鍵ペア $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ を $\mathcal{A}$ に返す。

TS-Rslease

T をオラクルへの入力とし、オラクルは時刻鍵 $S_T \leftarrow \mathsf{TS\text{-}Rslease}(params, ts_{priv}, T)$ を返す。ただし $T = T^*$ の場合は $\perp$ を返す。

Challenge

このオラクルへのクエリは 1 回のみ許される。$(pk^*, T^*, m_0^*, m_1^*)$ をオラクルへの入力とする。pk^* をチャレンジ key とする。オラクルは $b \leftarrow \{0, 1\}$ をランダムに決定しチャレンジ暗号文 $C^* \leftarrow \mathsf{Enc}(pk^*, T^*, m_b^*)$ を返す。ただし入力 T^* が TS-Rslease でクエリされている場合⊥を返す。

Decryption

(pk, T, C) をオラクルへの入力するとし、メッセージ $m = \mathsf{Dec}(sk, C, S_T)$ を返す。ただし $C = C^*$ または $C \Leftarrow C^*$ の場合は⊥を出力する。

Decision

$\mathcal{A}$ は b' を予想し出力する。

もし b' と b が等しければ $\mathcal{A}$ の勝利となる。$\mathcal{A}$ の勝利確率を次のように定義する。$Adv_{\Gamma,\mathcal{A}}(1^k) = 2Pr[b = b'] - 1$

定義 2：hTR-PRE 方式 Γ に対する任意の多項式時間アルゴリズム $\mathcal{A}$ の勝利確率 $Adv_{\Gamma,\mathcal{A}}(1^k)$ が negligible である時 Γ は Malicious User Security を満たすという。

3.3 プロトコル

本節では提案方式の詳細を書く。

— Setup

セキュリティパラメータ 1^k を入力とする。G, G_T を素数位数 p の群、$e : G \times G \rightarrow G_T$ とし、$g, g_2, g_3, h_1, h_2, h_3 \in G$、$s \leftarrow Z_p^*$ を選択する。$TS_{pub} = g^s, ts_{priv} = s$ とし、ハッシュ関数を以下のように選択する。

・ ハッシュ関数 $H : \{0,1\}^l \to G$
ハッシュ関数 H は pairwise independent な universal one-way hash function family [4][6] を用いる。$x \in \{0,1\}^l$ と $y \in G$ が与えられたとき、$H(x) = y$ を満たす H を効率よく見つけるアルゴリズムが存在する。
・ ハッシュ関数 $F : Z_p \to G$、ただしハッシュ関数 F は $F(y) = g_2^y \cdot g_3$ とする。$(g_2, g_3 \in G)$
・ ハッシュ関数 $H_1 : \{0,1\}^m \to Z_p$
・ ハッシュ関数 $H_2 : \{0,1\}^* \to Z_p$ $params = (g, g_2, g_3, h, h_1, h_2, h_3, H, F, H_1, H_2, TS_{pub})$ を公開パラメータとする。

— **KeyGeneration**
セキュリティパラメータ 1^k を入力とし、$x \in Z_p$ をランダムに選択。$pk = g^x, sk = x$ とする。

— **Re-EncryptionKeyGeneration**
$sk_i = x_i, sk_j = x_j$ を入力とし再暗号化鍵 $rk_{ij} = \frac{sk_i}{sk_j} \bmod p = \frac{x_i}{x_j} \bmod p$ とする。

— **TS-Release**
開示時刻 T とタイムサーバの秘密鍵 $ts_{priv} = s$ を入力とし $rT_1, rT_2, rT_3 \leftarrow Z_p^*$ をランダムに選択。開示時刻に対応する時刻鍵 S_T を以下の通り計算する。
$S_T = ((rT_1, (h_{rT_1}), (rT_2, (h_{rT_2}), (rT_3, (h_{rT_3}))$
$h_{rT_n} = (h_n \cdot g^{-rT_n})^{\frac{1}{(s-H(T))}}, (n \in \{1,2,3\})$

— **Encryption**
pk とメッセージ $m \in G_T$、開示時刻 T を入力とする。
・ ワンタイム署名の KeyGeneration アルゴリズムより $\mathsf{Gen}(1^k) \to (svk, ssk)$ を得る。$C_1 = svk$ とする。
・ 乱数 $r_1, r_2 \in Z_p$ を選択し次を計算する。
$C_2 = pk^{r_1}$, $C_3 = m \cdot e(g, H(svk))^{r_1} \cdot e(g, h_1)^{r_2})$
$C_4 = F(svk)^{r_1} = (g_2^{svk} \cdot g_3)^{r_1}$, $C_5 = h^{r_1}$
$C_6 = (g^{-H_1(T)} \cdot Ts_{pub})^{r_2}$, $C_7 = e(g,g)^{r_2}$
$C_8 = (e(e(g, h_2) \cdot e(g, h_3))^{\beta})^{r_2}$
$\beta = H_2(C_3, C_6, C_7)$
・ $(C_3, C_4, C_5, C_6, C_7, C_8)$ に対してワンタイム署名の Signature アルゴリズムを実行。
$\sigma \leftarrow \mathsf{Sig}(ssk, (C_3, C_4, C_5, C_6, C_7, C_8))$
・ $C = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, \sigma)$ を暗号文とする。

— **Re-Encryption**
暗号文 C_j と再暗号化鍵 rk_{ij} を入力とし次の計算を行う。
・ $C_2' = C_2^{rk_{ij}} = g^{x_i r_1}$
・ $\mathsf{CheckPRE}(C_j, pk_j) = 1$ ならば $C_i = (C_1, C_2', C_3, C_4, C_5, C_6, C_7, C_8, \sigma)$ として出力。そうでない場合は $\perp$ を出力。

— **Decryption**
秘密鍵 sk、公開鍵 pk、暗号文 C を入力とする。$\mathsf{CheckPRE}(C, pk) = 1$ かつ $\mathsf{CheckIBE}(C, pk) = 1$ ならば以下を計算する。そうでなければ$\perp$を出力する。
・ $e(C_6, h_{rT_1}) \cdot C_7^{rT_1} = e(g, h_1)^{r_2}$
・ $m = C_3 / \{e(C_2, H(C_1))^{1/sk} \cdot e(g, h_1)^{r_2}\}$ を出力。

方式で用いた CheckPRE アルゴリズムおよび CheckIBE アルゴリズムについて書く。
CheckPRE アルゴリズムを以下のように定義する。入力は暗号文 $C = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, \sigma)$ と公開鍵 pk である。
(1) $\mathsf{Ver}(C_1, (C_2, C_3, C_4, C_5, C_6, C_7, C_8), \sigma)$ を調べる。σ は $(C_3, C_4, C_5, C_6, C_7, C_8)$ に対する署名であり C_1 は検証鍵である。
(2) $e(C_2, F(C_1)) = e(pk, C_4), e(C_2, h) = e(pk, C_5)$ が成り立つか調べる
(3) 1,2 のどちらか一方でも成り立たなければ 0、そうでなければ 1 を出力。
CheckIBE アルゴリズムを以下のように定義する。入力は暗号文 $C = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, \sigma)$ と公開鍵 pk である。
(1) $C_8 = e(C_6, (h_2 \cdot g^{-rT_2})^{\frac{1}{s-H_1(T)}}, C_7^{rT_2+rT_3})$ が成り立つ場合は 1 を成り立たない場合は 0 を出力する。
提案した hTR-PRE は、C.H.PRE[7] と G.IBE[1] を組み合わせるアイデアに基づいて構成した。$C_1 = svk, C_2 = pk^{r_1}, C_4 = F(svk)^{r_1} = (g_2^{svk} \cdot g_3)^{r_1}, C_5 = h^{r_1}$ の要素は C.H.PRE の Enc アルゴリズムと同様であり、$C_6 = (g^{-H_1(T)} \cdot Tspub)^{r_2}, C_7 = e(g,g)^{r_2}, C_8 = (e(e(g, h_2) \cdot e(g, h_3))^{\beta})^{r_2}$ の要素は G.IBE の Enc アルゴリズムと同様である。$C_3 = m \cdot e(g, H(svk))^{r_1} \cdot e(g, h_1)^{r_2})$ において $m \cdot e(g, H(svk))^{r_1}$ の部分のみに注目すると、C.H.PRE の暗号文の要素の一部になり、また $m \cdot e(g, h_1)^{r_2}$ の部分に注目すると G.IBE の暗号文の要素の一部になる。

3.4 安全性証明

本節では提案した安全性定義に基づき hTR-PRE の安全性の証明を行う。

3.4.1 Malicious User Secrity

Malicious User Secrity を破る攻撃者 $\mathcal{A}$ を用いて GentryIBE の安全性を破るアルゴリズム $\mathcal{B}$ を構成する。

定理 1：IND-ID-CCA 安全な GentryIBE を用いて構成した hTR-PRE は Malicious User Secrity を満たす。

証明
挑戦者 $\mathcal{CH}$ は $g, h_1, h_2, h_3 \in G_1$ をランダムに選択する。$s \leftarrow Z_p^*$ をランダムに選択し、$ts_{prive} = s, TS_{pub} = g^s$ とする。$(g, h_1, h_2, h_3, TSpub, H_1, H_2)$ をシミュレータ $\mathcal{B}$ に

送る。

Setup

$\mathcal{B}$ は $g_2, g_3, h \in G$ をランダムに選択する。またワンタイム署名の KeyGeneration アルゴリズム $\mathsf{Gen}(1^k) \rightarrow (ssk^*, svk^*)$ を実行し $(g, g_2, g_3, h, h_1, h_2, h_3, e, H, F, H_1, H_2, TSpub, svk^*)$ を $\mathcal{A}$ に送る。

KeyGen

$\mathcal{A}$ がクエリすると $\mathcal{B}$ は $x \leftarrow Zp$ をランダムに選択し $(pk, sk) = (g^x, x)$ を $\mathcal{A}$ に返す。

TS-Release

$\mathcal{A}$ は TS-Release クエリとして時刻 T を $\mathcal{B}$ に送る。$\mathcal{B}$ は IND-ID-CCA ゲームにおける Extract クエリとして T をそのまま $\mathcal{CH}$ に送り, 対応する復号鍵（秘密鍵）d_T を手に入れる。$\mathcal{B}$ は TS-Release クエリの返答として $S_T = d_T$ を $\mathcal{B}$ に返す。

Challenge

$\mathcal{A}$ は 2 つの平文、時刻、任意の公開鍵のセット (m_0, m_1, pk^*, T^*) を $\mathcal{B}$ に送る。$\mathcal{B}$ は $r_1 \leftarrow Z_p$ をランダムに選択し $m'_0 = m_0 \cdot e(g, H(svk^*)), m'_1 = m_1 \cdot e(g, H(svk^*))^{r_1}$ を計算する。次に $\mathcal{B}$ は IND-ID-CCA ゲームにおける Challenge として $\mathcal{CH}$ に (m'_0, m'_1, T^*) を送る。$\mathcal{CH}$ は $b \leftarrow \{0,1\}$ と乱数 $r_2 \leftarrow Z_p$ を選択し以下を計算する。

$$C_3^* = m'_b \cdot e(g, h_1)^{r2},\ C_6^* = (g^{-H_1(T)} \cdot TS_{Pub})^{r_2},$$
$$C_7^* = e(g, g)^{r_2},\ C_8^* = (e(g, h_2) \cdot e(g, h_3)^{\beta})^{r_2}$$

$\mathcal{CH}$ は暗号文 $C = (C_3^*, C_6^*, C_7^*, C_8^*)$ を IND-ID-CCA のチャレンジ暗号文として $\mathcal{B}$ に送る。$\mathcal{B}$ は $C_1^* = svk^*, C_2^* = pk^{r_1}, C_4^* = F(svk^*)^{r_1}, C_5^* = h^{r_1}, \sigma = \mathsf{Sig}(ssk^*, (C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, C_8^*))$ を計算する。$\mathcal{B}$ は Challenge の返答として $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, C_8^*, \sigma^*)$ を $\mathcal{A}$ に送る。

Decryption

$\mathcal{A}$ は Decryption クエリとして公開鍵、暗号文、時刻 (pk, C, T) を $\mathcal{B}$ に送る。

$C_3 = m'_b \cdot e(g, h_1)^{r2}, C_6 = (g^{-H_1(T)} \cdot TS_{Pub})^{r_2},$
$C_7 = e(g, g)^{r_2}, C_8 = (e(g, h_2) \cdot e(g, h_3)^{\beta})^{r_2}$
$\sigma = \mathsf{Sig}(ssk^*, (C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, C_8^*))$ がすべて成り立つ場合 $C \Leftarrow C^*$ なので⊥を出力する。そうでない場合 $\mathcal{B}$ は以下の処理を行う。

(1) $\mathsf{CheckPRE}(pk, C) = 0$ または $\mathsf{CheckIBE}(pk, C) = 0$ の場合暗号文は正しく生成されていないので $\perp$ を返す。
(2) $e(C_6, h_{T1}) \cdot C_7^{rT1} = e(g, h_1)^{r_2}$, $C'_1 = \frac{C_3}{e(g, h_1)^{r_2}}$ を計算する。
(3) B は IND-ID-CCA ゲームにおける Decryption クエリとして $C^* = (C'_1, C_6, C_7, C_8)$ を $\mathcal{CH}$ に送り、復号結果 m をそのまま $\mathcal{A}$ に送る。

Decition

$\mathcal{A}$ は b' を予想し出力する。$\mathcal{B}$ は b' を受け取り、$b = b'$ ならば 1 を、そうでなければ 0 を出力する。

$\mathcal{A}$ が $C' = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C'_6, C'_7, C'_8, \sigma^*)$, $\mathsf{CheckPRE}(pk, C) = 1$, $\mathsf{CheckIBE}(pk, C) = 1$ をすべて満たす (pk', C') をクエリしてきた場合、$\mathcal{B}$ は Decryption オラクルをシミュレートすることが不可能である。しかし、これは IND-ID-CCA の安全性を破っていることになる。その他のシミュレーションについては失敗する事はないので、$\mathcal{A}$ の勝利確率が 1/2 に対して優位性を持つならば、$\mathcal{B}$ の成功確率も 1/2 に対して優位性を持つ。

3.4.2 Malicious TimeServer Security

定義した Malicious TimeServer Security を破る攻撃者 $\mathcal{A}$ を用いて mDBDH 問題を破るアルゴリズム $\mathcal{B}$ を構成する。

定理 2：mDBDH 仮定の下で hTR-PRE は Malicious TimeServer Security を満たす。

証明

$\mathcal{B}$ への入力は mDBDH の入力 (g, g^a, g^b, g^c, Q) となり、$\mathcal{B}$ のゴールは $Q = e(g, g)^{ab/c}$ が成り立つかジャッジすることである。アルゴリズム $\mathcal{B}$ は次のように構成する。

Setup

$\mathcal{B}$ はワンタイム署名の鍵生成アルゴリズム $\mathsf{Gen}(1^k) \rightarrow (ssk^*, svk^*)$ を実行する。次に $\omega, \alpha_1, \alpha_2 \in Z_p$ をランダムに選択し、$h = g^{c\omega}, g_2 = g^{\alpha_1}, g_3 = g^{-\alpha_1 svk^*} \cdot g^{c\alpha_2}$ を計算する。$s \leftarrow Z_p$ をランダムに選択し、タイムサーバの秘密鍵と公開鍵を $ts_{priv} = s, TS_{pub} = g^s$ とする。$\mathcal{B}$ は $H(svk*) = g^a$ となるようなハッシュ関数 H を選び、システムパラメータを $(p, h, h_2, h_3, ts_{priv}, TS_{pub}, g_{,2}, g_3, G, G_T, e, H_1, H_2, H, F)$ とする。

Uncorrupted key generation

$\mathcal{B}$ は乱数 $x \in Z_p$ を選び $pk = g^{cx}$ とし $\mathcal{A}$ に出力する。

Corrupted key generation

$\mathcal{B}$ は乱数 $x \in Z_p$ を選び $pk = g^x, sk = x$ とし $\mathcal{A}$ に出力する。

Re-Encryption key generation

pk_i, pk_j を入力とする。pk_i, pk_j ∈Uncorrupted または pk_i, pk_j ∈Corrupted の場合 $rk_{ij} = \frac{x_j}{x_i}$ として $\mathcal{A}$ に出力する。そうでない場合 $\perp$ を出力する。

Challenge

$\mathcal{A}$ はオラクルに (pk^*, m_0, m_1) を送信する $(pk^*$ ∈Uncorrupted$)$。$\mathcal{B}$ はランダムに $b \leftarrow \{0,1\}$ を選択し以下を計算する。$C_1^* = svk^*, C_2^* = (g^b)_i^x = pk_i^{b/c}, C_3^* = Q \cdot m_b \cdot e(g, h_1)^{r_2}, C_4^* = (g^b)^{\alpha_2} = (g_2^{svk*} \cdot g_3)^{b/c}, C_5^* = (g^b)^{\omega} = h^{b/c}, C_6^* = (g^{-H_1(T)} \cdot TS_{Pub})^{r_2}, C_7^* =$

$e(g,g)^{r_2}, C_8^* = (e(g,h_2) \cdot e(g,h_3)^{\beta})^{r_2}, \sigma^* = \mathsf{Sig}(ssk^*, (C_3, C_4, C_5, C_6, C_7, C_8))$ チャレンジ暗号文 $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, C_8^*, \sigma^*)$ を $\mathcal{A}$ に与える。

Decryption

(pk, C, T) を入力とする。$\mathsf{CheckPRE}(C, pk) = 0$ または $\mathsf{CheckIBE}(C, pk) = 0$ の場合暗号文は正しく生成されてないので $\mathcal{B}$ は $\perp$ を出力する。次に $C \Leftarrow C^*$ であるか確認をする。$C_1 = svk*, C_3 = Q \cdot m_b \cdot e(g,h_1)^{r_2}, C_4 = (g^b)^{\alpha_2}, C_5 = (g^b)^{\omega}, \sigma = \sigma^*$ を確認しすべて成り立つ場合、C は C^* を再暗号化して得られたもの、$C \Leftarrow C^*$ になるため $\perp$ を出力する。そのどちらでもない場合、$\mathcal{B}$ は次のような計算をしメッセージ m を出力する。

$t = \frac{C_4}{C_2^{\alpha_2/x}}, \lambda = \frac{1}{\alpha_1(C_1 - svk)}, \ m = \frac{C_3}{e(t^{\lambda}, H(C_1))e(g,h_1)^{r_2}}$

正しく生成された暗号文は $C_2 = pk^{r_1}, C_4 = F(C_1)^{r_1}$ となり、同じ乱数 r_1 を用いている。

$$t = \frac{F(C_1)^{r_1}}{pk^{r_1\alpha_2/x}} = \frac{g_2^{r_1 C_1} g_3^{r_1}}{pk^{\frac{r_1\alpha_2}{x}}} = \frac{(g^{\alpha_1})^{r_1 C_1}(g^{-\alpha_1 svk + c\alpha_2})^{r_1}}{(g^{cx})^{\frac{r_1\alpha_2}{x}}} = \frac{g^{r\alpha_1(C_1 - svk) + c\alpha_2}}{g^{r_1 c\alpha_2}} = g^{r_1\alpha_1(C_1 - svk)}$$

よって $C_1 = svk^*$ でないとき $t^{\lambda} = g^r$ になる。

Re-Encryption

(pk_i, pk_j, C_i) を入力とする。$\mathsf{CheckPRE}(C, pk) = 0$ または $\mathsf{CheckIBE}(C, pk) = 0$ の場合、暗号文 C_i は正しく生成されていないので $\perp$ を出力する。また、$i \in$Target かつ $j \in$Corrupt かつ $C \Leftarrow C^*$ つまり $C_1 = svk*, C_3 = Q \cdot m_b \cdot e(g,h_1)^{r_2}, C_4 = (g^b)^{\alpha_2}, C_5 = (g^b)^{\omega}, \sigma = \mathsf{Sig}(ssk^*, (C_3, C_4, C_5, C_6, C_7, C_8))$"の場合も $\perp$ を返す。そうでない場合は以下のように動作する。

(1) "$(i,j) \in$Uncorrupted, Target, Corrupted"または"$i \in$Uncorrupted かつ $j \in$Target"または"$j \in$Uncorrupted かつ $i \in$Target"の場合、$C_j = \mathsf{ReEnc}(x_j/x_i, C_i)$ を出力する。

(2) $i \in$Corrupted かつ $j \in$Uncorrupted の場合 B は $C_5^{x_j/\omega} = g^{crx_j} = pk_j^r = C_2'$ を計算し、$C_j = (C_1, C_2', C_3, C_4, C_5, C_6, C_7, C_8, \sigma)$ を出力する。

(3) $i \in$Uncorrupted かつ $j \in$Corrupted の場合、$t^{\lambda} = g^r$ より $g^{rx_j} = pk_j^r = C_2'$ を計算し $C_j = (C_1, C_2', C_3, C_4, C_5, C_6, C_7, C_8, \sigma)$ を出力する。

Decision

$\mathcal{A}$ は b' を予想し出力する。$\mathcal{B}$ は b' を受け取り、$b = b'$ ならば 1 を、そうでなければ 0 を出力する。

$\mathcal{A}$ が $C1 = svk^*$ となる暗号文をクエリする場合がある。チャレンジ暗号文 C^* を得た後、$\mathsf{CheckPRE}(C, pk) = 1$, $\mathsf{CheckIBE}(C, pk) = 1$、$C_1 = svk^*, C \neq C^*$ の全てをを満たす暗号文をクエリすると、Decryption、Re-Encryption オラクルはシミュレーションが不可能である。しかしそのような暗号文は $(C_3, C_4, C_5, C_6, C_7, C_8, \sigma) \neq (C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, C_8^*, \sigma^*)$ となっているためワンタイム署名を破られていることになる。$\mathcal{B}$ が mDBDH の問題を受け取ったならば mDBDH 問題を埋め込んだチャレンジ暗号文 C^* の分布と m_b をメッセージとした通暗号文 $C' = \mathsf{Enc}(pk^*, m_b^*, T^*)$ の分布は完全に一致する。よって $\mathcal{A}$ が $b' = b$ を正解する確率は、mDBDH の問題を埋め込んだ場合も、m_b を用いて通常の暗号化を行った場合も等しくなる。$\mathcal{B}$ が mDBDH の問題を受け取っていない場合は暗号文の要素 C_3 は b と独立して G_T 上に一様に分布するので m_b の情報は一切含んでいない。よって $\mathcal{A}$ が $b' = b$ を正解する確率は 1/2 となる。以上より mDBDH 仮定の下で hTR-PRE は Malicious TimeServer Security を満たす。

4. まとめ

hTR-PRE の構成法を示し安全性証明を行った。本研究では提案した hTR-PRE 方式に対し Malicious TimeServer Security と Malicious User Security の 2 つの安全性を定義し、安全性の証明を行った。Malicious TimeServer Security の安全性は mDBDH 仮定という計算量に基づく仮定に帰着し、Malicious User Security の安全性は G.IBE[3] の安全性 IND-ID-CCA に帰着する。提案した方式では G.IBE と C.H.PRE[7] を用いて方式を構成したが G.IBE 以外の IBE を用いることでも構成することが可能であると考えられる。しかし G.IBE と C.H.PRE は共通して DBDH 仮定を安全性の根拠としているので、安全性の面では最良であると考える。

参考文献

[1] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In EUROCRYPT, pages 457-473, 2005.

[2] Benoit Libert, Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy reencryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360—379. Springer, Heidelberg (2008).

[3] Craig Gentry. Practical identity-based encryption without random oracles. In Advances in Cryptology—EUROCRYPT 2006, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[4] Dan Boneh and Xavier Boyen. Efcient selective-ID secure identity-based encryption without random oracles. In EUROCRYPT '04, vol. 3027 of LNCS, pages 223-238, 2004.

[5] Keita Emura, Atsuko Miyaji, and Kazumasa Omote. A Timed-Release Proxy Re-encryption Scheme and Its Application to Fairly-Opened Multicast Communication. In ProvSec 2010, LNCS 6402, pp. 200—213, 2010.

[6] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In EUROCRYPT, vol 2656 of LNCS, pp. 255-271, 2003.

[7] Ran Canetti, Susan Hohenbergery. Chosen-Ciphertext Secure Proxy Re-Encryption. In ACM CCS 2007, pp.185—194. ACM Press, New York (2007).

[8] 小澤 賢佑 , 齊藤 泰一. Canetti-Hohenberger の multi-hop プロキシ再暗号化安全性モデルについて. 2013 年暗号と情報セキュリティシンポジウム,2013.