コンピュータセキュリティ 6－2
（1999. 7. 9）

岡本-田中鍵共有方式のなりすましに関する安全性

岡本 健† 多田 充† 岡本 栄司‡
{kenchan, mt}@jaist.ac.jp okamoto@cs.uwm.edu

†北陸先端科学技術大学院大学 情報科学研究科

Abstract

岡本-田中鍵共有方式の受動的攻撃 (passive attack) に対する安全性ついて数学的に厳密な考察が、満保-静谷によりなされた。しかしながら、攻撃者自らが生成した値を鍵共有の相手に送信し、なりすましを行なう能動的攻撃 (active attack) については、現在においても十分な議論がされていない。本論文では、まずそれぞれの攻撃について特徴を述べた後、具体的な攻撃方法をいくつか取り上げ、最終的に岡本-田中方式において なりすましをするには RSA を破る関数が必要であることを証明する。

The Rigorous Security for Okamoto-Tanaka ID-based Key Exchange Scheme against Active Attack

Takeshi Okamoto† Mitsuru Tada† Eiji Okamoto‡

†School of Information Science,
Japan Advanced Institute of Science and Technology

‡Center for Cryptography, Computer and Network Security(CCCNS),
University of Wisconsin, Milwaukee

Abstract

Okamoto-Tanaka ID-based key exhange scheme is based on the Diffie-Hellman key exchange scheme for key sharing, and which includes RSA-based authentication against impersonation. Hence, it is very important to consider the security on both key sharing and authentication. In 1998, the rigorous security against passive attack was studied by Mambo and Shizuya. However, the security for active attack such as intruder-in-the-middle attack has not been studied very well until now. In this paper, we will indicate some features on attacks at first, give concrete attacks, and finally prove the theorem that to impersonate the regular users, the attacker needs the function which can break the RSA scheme.

1 はじめに

研究の背景

岡本-田中鍵共有方式 [3] は、鍵共有に Diffie-Hellman 方式 [1] を用い、ユーザの認証には RSA 方式 [2] を用いた鍵共有方式であるが、安全性に関して数学的に厳密な考察は最近までなされていなかった。

1998 年に満保-静谷は関数間における帰着を用いて、岡本-田中方式の受動的攻撃 (passive attack) に対する考察を行なった [6]。これにより、岡本-田中方式の鍵共有に関しては Diffie-Hellman 方式と同程度に安全であることが証明されている。

しかしながら、中間侵入攻撃 (intruder-in-the-middle attack) に代表されるように、攻撃者自らが生成した値を鍵共有の相手に送信し、なりすましを行なう能動的攻撃 (active attack) については、現在においても十分な議論がされていない。これは、岡本-田中方式が鍵を共有する際に事前準備としてユーザ間で ある値を送信し、その後共有鍵を計算する one-path 方式であるため、能動的攻撃による安全性の考察が難しくなっていることが原因である。

前述の通り、岡本-田中方式の認証については RSA 方

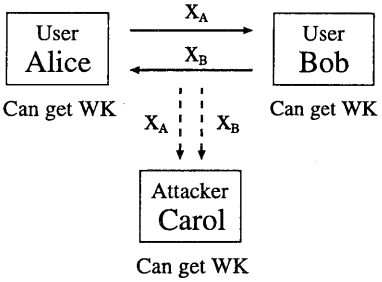

図 1: 受動的攻撃

式を用いている。このため RSA を破ることができればなりすましは成功する。しかし RSA を破る以外になりすましを行なう方法はない、という証明はされていない。すなわち、RSA を破ることが難しくても、効率的なアルゴリズムにより、なりすましを簡単に行なえる可能性が残されている。公開鍵暗号系に基づく方式で大切なのは、公開鍵（暗号鍵）から復号鍵が容易に計算できないという点であり、安全性について考察することは実用化を考える上でも大変重要なことである。

本研究では、この点について考察を行ない、なりすましをするには RSA を破る関数が必要であることを証明する。

岡本-田中鍵共有方式

この方式は次の三つのフェーズからなる。

センター情報生成フェーズ

認証局であるセンターは 2 つの素数 p , q および、n, g, e, d を生成する。ここで、$n = pq$ 、 g は Z_p^* と Z_q^* の原始元、e, d は $e \in Z_{\lambda(n)}^*$, $\lambda(n) = \mathrm{lcm}\,(p-1, q-1)$, $ed = 1 \pmod{\lambda(n)}$ を満たす。

ユーザ加入フェーズ

ID_i を ユーザ i の ID 情報とする。センターは

$$s_i = ID_i^{-d} \pmod n$$

となるユーザ i の秘密情報を生成する。

センターは (n, e, g, ID_i) を公開し、s_i を安全な通信路を用いてユーザ i に配布する。

ユーザ鍵生成フェーズ

ユーザ Alice と Bob の間で、ワーク鍵を生成するものとする。このとき、Alice は乱数 r_A を生成し、

$$x_A = g^{r_A} \cdot s_A \pmod n$$

を相手ユーザ Bob に送る。同様に Bob は乱数 r_A を生成し、

図 2: 能動的攻撃

$$x_B = g^{r_B} \cdot s_B \pmod n$$

を相手ユーザ Alice に送る。Alice は送られた x_B から

$$WK_{AB} = (ID_B \cdot x_B^e)^{r_A} \pmod n$$

を計算する。同様に Bob は送られた x_A から、

$$WK_{BA} = (ID_A \cdot x_A^e)^{r_B} \pmod n$$

を計算する。

以下の式を見ればわかるように、ワーク鍵 WK_{AB} と WK_{BA} は 互いに等しいので、Alice と Bob はこの鍵を共有鍵として用いることができる。

$$\begin{aligned} WK_{AB} &= (ID_B \cdot x_B^e)^{r_A} \\ &= (ID_B \cdot (g^{r_B} \cdot s_B)^e)^{r_A} \\ &= (ID_B \cdot ID_B^{-e \cdot d} \cdot g^{e \cdot r_B})^{r_A} \\ &= g^{e \cdot r_A \cdot r_B} \\ &= WK_{BA} \pmod n \end{aligned}$$

2 攻撃の種類とその特徴

岡本-田中方式を破る攻撃について考察するため、攻撃の種類を受動的攻撃 (passive attack) と能動的攻撃 (active attack) に分類し、それぞれを破る関数について述べる。また本論文では Alice と Bob を正規ユーザとし、攻撃者を Carol とする。

2.1 受動的攻撃

プロトコル スタイル

Step 1 Alice は $x_A = g^{r_A} \cdot s_A \pmod n$ を Bob に送る。

Step 2 Bob は $x_B = g^{r_B} \cdot s_B \pmod n$ を Alice に送る。

Step 3 Carol は 関数 OT を用いて 共有鍵 $WK = (ID_B \cdot x_B{}^e)^{r_A} = (ID_A \cdot x_A{}^e)^{r_B} = g^{e r_A r_B} \pmod n$ を計算する。

この攻撃の安全性については、満保、静谷により 、関数 OT を計算することの難しさは Diffie-Hellman 鍵共有方式 [1] を破る難しさと同程度であることが証明されている [6] 。

2.2 能動的攻撃

プロトコル スタイル

Step 1 Alice は $x_A = g^{r_A} \cdot s_A \pmod n$ を Bob に送る。

Step 2 Carol は 関数 Imp-Sp2を用いて x_C を計算し、Alice に送る。

Step 3 Carol は 関数 Imp-Sp3を用いて 共有鍵 $WK = (ID_B \cdot {x_C}^e)^{r_A} \pmod n$ を計算する。

それぞれの関数 Imp-Sp2 、Imp-Sp3 は次のような特徴を持つ。

- Imp-Sp2 は $(n, e, g, ID_A, ID_B, x_A)$ を入力値 として、「ある項 (term)」 x_C を出力する関数である。
- Imp-Sp3 は $(n, e, g, ID_A, ID_B, x_A, x_C)$ を入力値として、 $WK = (ID_B \cdot {x_C}^e)^{r_A} \pmod n$ を出力する関数である。

能動的攻撃の安全性について考察が難しい理由は、攻撃者自らが x_C の項を設定できる点にある。つまり、x_C をどのような項にするかによって、関数 Imp-Sp2、Imp-Sp3 の難しさが変化する。もし、x_C の設定により両方の関数が多項式時間で計算可能ならば、岡本-田中方式は能動的攻撃により破れることになる。

また、Imp-Sp2 、Imp-Sp3 を計算する難しさが相対的に異なる場合、岡本-田中方式を破る難しさは、2 つの関数の関係により次のように変化する。

順序関係がある場合

難しい方の関数に依存する。例えば、Imp-Sp2を計算する難しさが RSA 方式を破る難しさと等価であり、Imp-Sp3については Diffie-Hellman 方式と等価であるならば、岡本-田中方式を破ることの難しさは Diffie-Hellman 方式を破る難しさと等価である。

順序関係がない場合

両方の関数に依存する。例えば、Imp-Sp2を計算する難しさが 素因数分解を破る難しさと等価であり、Imp-Sp3 については Diffie-Hellman 方式と等価であるならば、岡本-田中方式を破ることの難しさは 素因数分解と Diffie-Hellman 方式の両方を破る難しさと等価である。

3 準備

本論文では、関数の複雑さの関係を比較するために帰着という概念を用いる。簡単にいえば、関数 G を計算するサブルーチン（オラクル）を用いて、関数 F が計算できるとき、関数 F は 関数 G に帰着すると言う。以下で述べる 2 種類の帰着を、本論文の内容に沿った形式で述べるが、より一般的で厳密な定義については、例えば [4] などに記述されている。

定義 3.1 [多項式時間 many-one 帰着] 関数 F, G に関して、もし任意の x に対して $F(x) = h_1(G(h_2(x)))$ となるような多項式で計算可能な関数 h_1, h_2 が存在するならば、F は G に多項式時間 many-one 帰着するといい、$F \leq_m^{FP} G$ と表記する。逆についてもいえるならば、F は G に多項式時間 many-one 帰着において等価であるといい、$F \equiv_m^{FP} G$ と表記する。

定義 3.2 [多項式時間 Turing 帰着] 関数 F, G に関して、もし任意の x に対し $|x|$ の多項式回数以内（ここでは j 回とする）のサブルーチン および 多項式で計算可能な関数 h_i, $(1 \leq i \leq j-1)$ を用いて $F(x) = h_j\left(\circ_{i=0}^{j-1}(G \circ h_i)\right)(x)$ となるならば、F は G に多項式時間 Turing 帰着するといい、$F \leq_T^{FP} G$ と表記する。逆についてもいえるならば、F は G に多項式時間 Turing 帰着において等価であるといい、$F \equiv_T^{FP} G$ と表記する。

次に、RSA 公開鍵暗号系、 法が n である Diffie-Hellman 鍵共有方式、それを応用した 拡張 Diffie-Hellman 鍵共有方式 を破るための関数を定義する。

定義 3.3 [RSA 公開鍵暗号系] $\mathtt{RSA}(n, e, y)$ は入力を $n \in N_{>1}$, $e \in Z_{\lambda(n)}^*$, $y \in Z_n^*$ としたとき、$y = x^e \pmod n$ なる $x \in Z_n^*$ が存在するとき、x を出力する。

定義 3.4 [Diffie-Hellman 鍵共有方式] $\mathtt{DH}(n, g, y_A, y_B)$ は入力を $n \in N_{>1}$, $g \in Z_n^*$, $y_A \in Z_n^*$, $y_B \in Z_n^*$ としたとき、$K = g^{ab} \pmod n$, $y_A = g^a \pmod n$, $y_B = g^b \pmod n$ なる $K \in Z_n^*$ が存在するとき、K を出力する。

定義 3.5 [拡張 Diffie-Hellman 鍵共有方式] $\mathtt{EDH}(n, g, \tilde{y}_A, \tilde{y}_B)$ は入力を $n \in N_{>1}$, $g \in Z_n^*$, $\tilde{y}_A \in Z_n^*$, $\tilde{y}_B \in Z_n^*$ としたとき、$\tilde{K} = \tilde{y}_B^{\tilde{a}} \pmod n$, $\tilde{y}_A = g^{\tilde{a}} \pmod n$ なる $\tilde{K} \in Z_n^*$ が存在するとき、$\tilde{K}$ を出力する。

関数 DH と EDH は、定義から明らかに $\mathtt{DH} \leq_m^{FP} \mathtt{EDH}$ となる。

最後に 岡本-田中方式のなりすましについて定義する。

定義 3.6 [岡本-田中方式のなりすまし] 岡本-田中方式において攻撃者が能動的攻撃により正規ユーザとの間で用いる共有鍵を計算（あるいは入手）し、通信を行なう攻撃を 岡本-田中方式のなりすましと呼ぶ。

4 各種の攻撃方法

なりすまし攻撃により 岡本-田中方式を破るために、いくつかのプロトコルを考える。このプロトコルの中で共有鍵 WK 等の出力値を計算する関数を定義し、この関数の難しさについて考察する。

4.1 攻撃 1

Carol が Alice に乱数を送ることにより、共有鍵 WK を計算する難しさについて考える。

プロトコル 1

Step 1 Alice は $x_A = g^{r_A} \cdot (ID_A)^{-d} \pmod n$ を Carol に送る。

Step 2 Carol は 乱数 $\gamma_C \in Z_n^*$ を生成し、Alice に送る。

Step 3 Carol は関数 Imp1-Sp3を用いて共有鍵 $WK = (ID_B \cdot \gamma_C^e)^{r_A} \pmod n$ を計算する。

定義 4.1 Imp1-Sp3$(n, e, g, ID_A, ID_B, x_A, \gamma_C)$ は入力を $n \in N_{>1}$, $e \in Z_{\lambda(n)}^*$, $g \in Z_n^*$, $ID_A \in Z_n^*$, $ID_B \in Z_n^*$, $x_A \in Z_n^*$, $\gamma_C \in Z_n^*$ としたとき、 $WK = (ID_B \cdot \gamma_C^e)^{r_A} \pmod n$, $x_A = g^{r_A} \cdot (ID_A)^{-e^{-1}} \pmod n$, $ee^{-1} = e^{-1}e = 1 \pmod{\lambda(n)}$ となる $WK \in Z_n^*$ が存在するとき、 WK を出力する。

定理 4.2 Imp1-Sp3 $\equiv_m^{FP}$ EDH.

証明

1. Imp1-Sp3 $\leq_m^{FP}$ EDH:

$$\text{Imp1-Sp3}(n, e, g, ID_A, ID_B, x_A, \gamma_C) = \text{EDH}(n, g^e, ID_A \cdot x_A^e, ID_B \cdot \gamma_C^e).$$

2. EDH $\leq_m^{FP}$ Imp1-Sp3:

$$\text{EDH}(n, g, \tilde{y}_A, \tilde{y}_B) = \text{Imp1-Sp3}(n, 1, g, 1, \tilde{y}_B, \tilde{y}_A, 1).$$

■

このことは Carol が EDH を持っていれば、どのような値を Alice に送っても、必ず共有鍵を計算できることを意味する。

4.2 攻撃 2

センターの代わりに Carol が Alice の秘密情報 s_A を計算し、その後は通常のプロトコルに従って共有鍵 WK を計算する難しさについて考える。

プロトコル 2

Step 1 Alice は $x_A = g^{r_A} \cdot (ID_A)^{-d} \pmod n$ を Carol に送る。

Step 2 Carol は 乱数 $r_C \in Z_n^*$ を生成し、関数 Imp2-Sp2を用いて $x_C = g^{r_C} \cdot (ID_B)^{-d} \pmod n$ を計算し、Alice に送る。

Step 3 Carol は 共有鍵 $WK = (ID_B \cdot x_C^e)^{r_A} = g^{er_Ar_C} \pmod n$ を計算する。

定義 4.3 Imp2-Sp2(n, e, g, ID_B, r_C) は入力を $n \in N_{>1}$, $e \in Z_{\lambda(n)}^*$, $g \in Z_n^*$, $ID_B \in Z_n^*$, $r_C \in Z_n^*$, としたとき、 $x_C = g^{r_C} \cdot ID_B^{-e^{-1}} \pmod n$, $ee^{-1} = e^{-1}e = 1 \pmod{\lambda(n)}$ となる $x_C \in Z_n^*$ が存在するとき、 x_C を出力する。

定理 4.4 Imp2-Sp2 $\equiv_m^{FP}$ RSA.

証明

1. Imp2-Sp2 $\leq_m^{FP}$ RSA:

$$\text{Imp2-Sp2}(n, e, g, ID_B, r_C) = \text{RSA}(n, e, g^{er_C}/ID_B).$$

2. RSA $\leq_m^{FP}$ Imp2-Sp2:

$$\text{RSA}(n, e, y) = \text{Imp2-Sp2}(n, e, 1, 1/y, 1).$$

■

このことは、 Carol がセンターの代わりに Alice の秘密情報を計算し、 これに Carol が生成した乱数を公開されている原始元にべき乗したものを掛けて Alice に送れば、RSA のみで共有鍵が計算できることを意味する。

4.3 攻撃 3

攻撃 2 と同様、 Carol が センターの代わりに Alice の秘密情報 s_A を計算するが、その後は通常のプロトコルと異なる方法で共有鍵 WK を計算する難しさについて考える。

プロトコル 3

Step 1 Alice は $x_A = g^{r_A} \cdot (ID_A)^{-d} \pmod n$ を Carol に送る。

Step 2 Carol は 乱数 $r_C \in Z_n^*$ を生成し、$g \neq \tilde{g}_C \pmod n$ となる定数 $\tilde{g}_C \in Z_n^*$ を選択する。また関数 Imp3-Sp2を用いて $x_C = \tilde{g}_C^{r_C} \cdot (ID_B)^{-d} \pmod n$ を計算し、Alice に送る。

Step 3 Carol は関数 Imp3-Sp3を用いて共有鍵 $WK = (ID_B \cdot x_C^e)^{r_A} = \tilde{g}_C^{er_Ar_C} \pmod n$ を計算する。

定義 4.5 Imp3-Sp2$(n, e, \tilde{g}_C, ID_B, r_C)$ は入力を $n \in N_{>1}$, $e \in Z_{\lambda(n)}^*$, $\tilde{g}_C \in Z_n^*$, $ID_B \in Z_n^*$, $r_C \in Z_n^*$, としたとき、 $x_C = \tilde{g}_C^{r_C} \cdot ID_B^{-e^{-1}} \pmod n$, $ee^{-1} = e^{-1}e = 1 \pmod{\lambda(n)}$ となる $x_C \in Z_n^*$ が存在するとき、 x_C を出力する。

定理 4.6 Imp3-Sp2 $\equiv_m^{FP}$ RSA.

証明

定理 4.4 より明らか。

■

定義 4.7 Imp3-Sp3$(n, e, g, \tilde{g}_C, ID_A, ID_B, x_A, x_C, r_C)$ は入力を $n \in N_{>1}$, $e \in Z_{\lambda(n)}^*$, $g \in Z_n^*$, $\tilde{g}_C \in Z_n^*$, $ID_A \in Z_n^*$, $ID_B \in Z_n^*$, $x_A \in Z_n^*$, $x_C \in Z_n^*$, $r_C \in Z_n^*$, としたとき、 $WK = (ID_B \cdot x_C{}^e)^{r_A} = \tilde{g}_C^{er_Ar_C} \pmod n$, $x_A = g^{r_A} \cdot (ID_A)^{-e^{-1}} \pmod n$, $x_C = \tilde{g}_C^{r_C} \cdot (ID_B)^{-e^{-1}} \pmod n$, $ee^{-1} = e^{-1}e = 1 \pmod{\lambda(n)}$ となる $WK \in Z_n^*$ が存在するとき、 WK を出力する。

定理 4.8 Imp3-Sp3 $\equiv_m^{FP}$ DH.

証明

1. Imp3-Sp3 $\leq_m^{FP}$ DH:

$$\text{Imp3-Sp3}(n, e, g, \tilde{g}_C, ID_A, ID_B, x_A, x_C, r_C) = \text{DH}(n, \tilde{g}^e, \tilde{g}_C^{er_C}, ID_A \cdot x_A^e).$$

2. DH $\leq_m^{FP}$ Imp3-Sp3:

$$\mathrm{DH}(n,g,y_A,y_B) = \mathrm{Imp3\text{-}Sp3}(n,1,g,g,1,1,y_A,y_B,1).$$

■

このことは、 Carol がセンターの代わりに Alice の秘密情報を計算し、これに Carol が生成した乱数を公開されている原始元以外の定数にべき乗したものを掛けて Alice に送れば、 共有鍵を計算するのに DH が必要であることを意味する。

5 安全性の証明

準備として原始帰納的関数の集合を $\mathcal{PRF}$ と表記する。また、

$\mathcal{S}$: 変数または定数の集合,
$\mathcal{F}$: 関数のクラス

としたとき、

$$\{\mathcal{S},\mathcal{F}\} := \{f(x) \mid x \in \mathcal{S}, f \in \mathcal{FP}^{\mathcal{F}}\}$$

と定義する。

証明の方法として、我々は、2.2 章で示したプロトコルの Step2 に注目する。この時、 Carol が Imp-Sp2 を用いて計算する x_C は、

$$x_C \in \left\{\mathcal{I},\mathcal{PRF}/\equiv_T^{FP}\right\},$$

$$\mathcal{I} := \left\{n,e,g,ID_l(l=A,B,C,\ldots),x_A,const\right\},$$

$const$: 定数の集合

と表すことができる。前述の通り、 Carol は Alice に 送る 項 x_C をどのように設定するかによって x_C を計算する難しさは変化するが、ここでは、 岡本-田中方式を破るすべての 項 $x_{Ci} \in \left\{\mathcal{I},\mathcal{PRF}/\equiv_T^{FP}\right\}, (i=1,2,\ldots)$ を列挙し、この項を計算する関数が次の 2 つの条件のうち、どちらに属するかを考察する。

条件 1 Step2 が終了した時点において、
$WK_i = (ID_B \cdot x_C^e)^{r_A} = g^{er_Ab_i} \pmod n$
となる $b_i \in Z_n$が (Carol にとって) 既知となるとき

条件 2 それ以外のとき

以後、 2 つの条件についてプロトコルを作成し、その中で用いられる関数のペアの難しさについて考察する。

条件 1 を満たす場合

プロトコル 4

Step 1 Alice は $x_A = g^{r_A} \cdot s_A \pmod n$ を Carol に送る。

Step 2 Carol は関数 Imp4-Sp2$_i$ を用いて、条件 1 を満たす $x_{Ci} \in \left\{\mathcal{I},\mathcal{PRF}/\equiv_T^{FP}\right\}$ を計算し Alice に送る。

Step 3 Carol は共有鍵 $WK = (ID_A \cdot x_A^e)^{b_i} = g^{er_Ab_i} \pmod n$ を計算する。

上記のプロトコルの Step3 において、$WK_i = (ID_A \cdot x_A^e)^{b_i} = g^{er_Ab_i} \pmod n$ は多項式時間で計算できることがわかっている。よって、ここでは 関数 Imp4-Sp2$_i$ を計算する難しさのみ考察すればよい。また、この関数の出力値 x_{Ci} は、

$$\begin{aligned} & (ID_B \cdot x_{Ci}^e)^{r_A} = g^{er_Ab_i} \\ \Leftrightarrow \quad & x_{Ci}^e = g^{eb_i} \cdot ID_B^{-1} \\ \Leftrightarrow \quad & x_{Ci} = g^{b_i} \cdot ID_B^{-d} \pmod n \end{aligned}$$

となることに注意 (付録参照)。このため関数 Imp4-Sp2$_i$ は次のように定義できる。

定義 5.1 Imp4-Sp2$_i$ (n,e,g,ID_A,ID_B,x_A) は入力を $n \in N_{>1}$, $e \in Z_{\lambda(n)}^*$, $g \in Z_n^*$, $ID_A \in Z_n^*$, $ID_B \in Z_n^*$, $x_A \in Z_n^*$, としたとき、$ee^{-1} = e^{-1}e = 1 \pmod{\lambda(n)}$, $x_A = g^{r_A} \cdot s_A \pmod n$, $x_{Ci} = g^{b_i} \cdot ID_B^{-d} \pmod n$, $b_i \in Z_n$, $f_i(x_{C_i}) = b_i$, $f_i \in \mathcal{FP}$, となる f_i, $x_{Ci} \in Z_n^*$ が存在するとき、 f_i, x_{Ci} を出力する。

定理 5.2 すべての i において、 RSA $\leq_T^{FP}$ Imp4-Sp2$_i$ が成り立つ。

証明

以下に RSA から Imp4-Sp2$_i$ への帰着のアルゴリズムを示す。

```
% Algorithm for reducing RSA to Imp4-Sp2_i
input (n, e, y)
begin
z_i := Imp4-Sp2_i(n, e, 1, 1, 1/y, 1);
x := P_2(z_i);  %  x = y^d
if x = ⊥
    then output "undedined"
    else output x
end-if
end.
```

ここで、 $P_k : \cup_{j=1}^{\infty} N^j \to N$ は射影関数であり、$P_k(x_1,x_2,x_3,\ldots,x_m) = x_k$ となる。

■

条件 2 を満たす場合

プロトコル 5

Step 1 Alice は $x_A = g^{r_A} \cdot s_A \pmod n$ を Carol に送る。

Step 2 Carol は関数 Imp5-Sp2$_i$ を用いて、条件2を満たす $x_{Ci} \in \left\{ \mathcal{I}, \mathcal{PRF} / \equiv_T^{FP} \right\}$ を計算し、Alice に送る。

Step 3 Carol は関数 Imp5-Sp3$_i$ を用いて共有鍵 $WK_i = (ID_B \cdot x_{Ci}^e)^{r_A} \pmod n$ を計算する。

定義 5.3 Imp5-Sp2$_i$ $(n, e, g, ID_A, ID_B, x_A)$ は入力を $n \in N_{>1}$, $e \in Z^*_{\lambda(n)}$, $g \in Z_n^*$, $ID_A \in Z_n^*$, $ID_B \in Z_n^*$, $x_A \in Z_n^*$ としたとき、$x_{Ci} \in \left\{ \mathcal{I}, \mathcal{PRF} / \equiv_T^{FP} \right\}$, $ee^{-1} = e^{-1}e = 1 \pmod{\lambda(n)}$, $x_A = g^{r_A} \cdot (ID_A)^{-e^{-1}} \pmod n$, $ee^{-1} = e^{-1}e = 1 \pmod{\lambda(n)}$ となる $x_{Ci} \in Z_n^*$ が存在するとき、x_{Ci} を出力する。

定義 5.4 Imp5-Sp3$_i$ $(n, e, g, ID_A, ID_B, x_A, x_{Ci})$ は入力を $n \in N_{>1}$, $e \in Z^*_{\lambda(n)}$, $g \in Z_n^*$, $ID_A \in Z_n^*$, $ID_B \in Z_n^*$, $x_A \in Z_n^*$, $x_{Ci} \in Z_n^*$, としたとき、$WK_i = (ID_B \cdot x_{Ci}{}^e)^{r_A} \pmod n$, $x_A = g^{r_A} \cdot (ID_A)^{-e^{-1}} \pmod n$, $x_{Ci} \in \left\{ \mathcal{I}, \mathcal{PRF} / \equiv_T^{FP} \right\}$, $ee^{-1} = e^{-1}e = 1 \pmod{\lambda(n)}$ となる $WK_i \in Z_n^*$ が存在するとき、WK_i を出力する。

次の定理が導ける。

定理 5.5 すべての i において、RSA $\leq_m^{FP}$ Imp5-Sp3$_i$ $\circ$ Imp5-Sp2$_i$ が成り立つ。

証明

$$\mathtt{RSA}(n, e, y) = \mathtt{Imp5\text{-}Sp3}_i \circ \mathtt{Imp5\text{-}Sp2}_i(n, e, y^e, 1, y, y)$$

■

定理 5.2 および 定理 5.5 から 次のことがいえる。

系 5.6 岡本-田中鍵共有方式において、なりすましをするには RSA を破る関数が必要である。

6 むすび

岡本-田中鍵共有方式のなりすましに関する安全性について考察を行なった。本論文では、まず具体的な攻撃方法をいくつか取り上げ、最終的には関数の帰着を用いて安全性を考察している。これにより岡本-田中鍵共有方式においてなりすましをするには RSA を破る関数が必要であることを証明した。

現在、RSA を多項式時間で解くアルゴリズムは知られていない。しかし定理 5.5 より、もし2つの関数がどちらも多項式時間で計算が可能ならば、$\mathcal{FP}$ に属する関数は合成に関して閉じているという事実から、RSA は多項式時間で破ることができる。逆に $\mathcal{FP}$ に属する2つの関数が存在しないならば、これは $\mathcal{P} \neq \mathcal{NP}$ を意味する。従ってこの証明は、少なくとも $\mathcal{P} \neq \mathcal{NP}$ を証明する程度の困難さを伴う。

今後の課題として今回行なった証明方法の検討および known key attack など、今回取り上げた以外の攻撃による安全性の証明等が考えられる。

謝辞

本研究に関して有益な助言を頂いた東北大学の静谷啓樹先生、満保雅浩先生、JAIST の宮地充子先生 および日立製作所の坂崎尚生氏に心から感謝致します。

参考文献

[1] W. Diffie and M. E. Hellman, "New directions in cryptography", *IEEE Trans. Information Theory*, vol. IT-22, pp.644-654, 1976.

[2] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystem", *Communications of ACM*, vol. 21, pp.120-126, 1978.

[3] E. Okamoto and K. Tanaka, "Key distribution system based on identification information", *IEEE J. Selected Areas in Communications*, Vol.7, pp.481-485, 1989.

[4] J. L. Balcázar, J. Díaz and J. Gabarró, "Structural Complexity I", EATCS Monograph on Theoretical Computer Science 11, Springer-Verlag, 1988.

[5] K. Sakurai and H. Shizuya, "Relationships among the computational powers of breaking discrete log cryptosystems", Proc. *Eurocrypt'95*, LNCS 921, Springer-Verlag, pp.341-355, 1995.

[6] M. Mambo and H. Shizuya, "A Note on the Complexity of Breaking Okamoto-Tanaka ID-Based Key Exchange Scheme", Proc. *PKC'98*, LNCS 1431, Springer-Verlag, pp.258-262, 1998.

[7] T. Oomi, H.Shizuya and T.Nishizeki "On the Complexity of Language Associated with Discrete Log Cryptosystems", Technical Report of IEICE, ISEC 97-32, pp.31-42, 1997.

付録

プロトコル4で示した関数 Imp4-Sp2$_i$ の出力値 x_{Ci} が存在する可能性について考察する。r_A が Z_n から一様に選ばれた場合、$p = 2p' + 1$, $q = 2q' + 1$, $p', q' \in N_{\text{prime}}$ とすると、同値関係 ($\Leftrightarrow$) が成り立つ確率は、

$$\frac{1}{n}\left(\varphi(2p'q')\frac{n}{\lambda(n)}\right) = \frac{1}{n}\left(\frac{(p'-1)(q'-1)pq}{2p'q'}\right)$$

$$= \frac{1}{2}\left(1 - \frac{1}{p'}\right)\left(1 - \frac{1}{q'}\right) \approx \frac{1}{2}$$

となる。