

## 権利管理システムへの権利貸与機能の適用について

山崎, 知美  
九州大学大学院システム情報科学府

井上, 創造  
九州大学附属図書館

中村, 徹  
九州大学大学院システム情報科学府

野原, 康伸  
九州大学大学院システム情報科学府

他

<https://hdl.handle.net/2324/8307>

---

出版情報：マルチメディア, 分散, 協調とモバイル(DICOM)シンポジウム論文集. 2007, pp.251-257,  
2007-07-04

バージョン：

権利関係：



# 権利管理システムへの権利貸与機能の適用について

山崎 知美<sup>†</sup> 井上 創造<sup>††</sup> 中村 徹<sup>†</sup>  
野原 康伸<sup>†</sup> 安浦 寛人<sup>†††</sup>

本論文では、既存の権利管理システムに変更を加えることなく貸与利用機能を付加させる方法を提案する。提案手法は利用者と既存システムの間で権利利用時にのみ代理処理をするプロキシを導入させる方法であるが、プロキシを用いる場合は一般にプロキシの信頼性が問題となる。そこで提案プロトコルにおける問題の影響を明らかにし、プロキシの信頼性を実現するための方法をプロキシの分類毎に検討する。特に利用者の領域にプロキシを置いた場合に、プロキシの信頼性を被貸与者に委託する方法を提案する。

## Applying Delegation Functionality to Privilege Management Systems

TOMOMI YAMASAKI,<sup>†</sup> SOZO INOUE,<sup>††</sup>  
TORU NAKAMURA,<sup>†</sup> YASUNOBU NOHARA<sup>†</sup>  
and HIROTO YASUURA<sup>†††</sup>

### 1. はじめに

各種サービスにおいては、権限を他人に制限付きで複製や委譲をして利用してもらう機能が求められることが多い。この機能を貸与利用機能と呼ぶ。文献<sup>4),5)</sup>においては貸与をサービスを介さずに、利用者間だけで行うことができる方式が提案された。しかし、この方式を含め、貸与利用機能を、既存のシステムに付加する方法については議論されていない。

既存システムに改造を加えることは多くの場合、手間と費用がかかるため、本論文では既存システムにできるだけ手を加えずに貸与利用機能を追加することを考える。その際に課題となる点は以下の2点である。

- (1) 既存システムが外部に公開するインターフェースは、ID とパスワードの入力といったように、限られている。
- (2) 利用者は追加や削除により、頻繁に変更することが多いが、利用者を知る機能は既存システム

は外部に公開しないことが多い。

本論文では、利用者と既存システムの間で代理処理をするプロキシを導入させることにより、上記課題を解決する貸与利用機能を付加させる方法を3節において提案する。提案する方法は、プロキシが貸与時の情報を記憶しておく必要のないオフライン型に基づくものである。

また、提案プロトコルの安全性を検証し、プロキシを用いない場合には問題とならなかった問題の影響を明らかにする。具体的には、プロキシの貸与者への成りすまし、プロキシによる改ざん、プロキシによる不正な再利用、貸与者の否認において問題が生じることを示す。

このためプロキシが信頼できることを前提とするか、プロキシの行動を監視する必要があるが、4節において、プロキシの分類毎に、これらの実現方法を議論する。特に利用者の領域にプロキシを置いた場合に、プロキシの行動を被貸与者が監視することで、貸与者はプロキシを信頼しなくても、被貸与者が監視することに信頼を置けばよい方法を提案する。この方法は、プロキシの信頼を被貸与者に委託したものであるといえ、現実の応用において、貸与者と被貸与者間でのみ信頼できる場合には有用であると考えられる。

<sup>†</sup> 九州大学大学院システム情報科学府  
Graduate School of Information Science and Electrical  
Engineering, Kyushu University

<sup>††</sup> 九州大学附属図書館  
Library, Kyushu University

<sup>†††</sup> 九州大学大学院システム情報科学研究院  
Faculty of Information Science and Electrical Engineering,  
Kyushu University

## 2. 貸与利用プロトコル

この節では、既存の貸与利用プロトコルおよび、既存の単純なサービスのモデルを述べる。貸与利用プロトコルには、貸与時に貸与元の利用者とサービスの間での通信を要する方式と、文献<sup>(4,5)</sup>で提案された、貸与元と貸与先の利用者の間での通信のみで良い方式がある。以下では前者をオンライン貸与利用プロトコル、後者をオフライン貸与利用プロトコルと呼ぶ。この節では、そのそれぞれのプロトコルを述べる。ただしこの節では単純のため、盗聴によるなりすましや不正な再利用を防ぐための暗号化と認証のための手続きを省略して記述する。また、貸与利用機能を持たない既存のサービスにこの機能を付加する際の要求をまとめる。

以下では、サービスを  $s$ 、利用者の集合を  $U$  とする。また、各利用者  $a \in U$  は、その識別子  $i_a$  を与えられているものとする。

両プロトコルに共通して、サービス  $s$  は認可関数  $F_s : U \rightarrow \{1, 0\}$  を持つ。例えば  $F_s(a) = 1, a \in U$  であれば、利用者  $a$  はサービス  $s$  を受けることができるという意味である。

以下では、

$$X : W$$

または

$$X \rightarrow Y : Z$$

というプロトコルの記述方法を用いる。前者は  $X$  が動作  $W$  をするという意味であり、後者は  $X$  が  $Y$  に  $Z$  を渡す事を意味する。

### 2.1 オンライン貸与利用プロトコル

オンライン貸与利用プロトコルは以下のように書くことができる。

利用者  $a$  から  $b$  への貸与手順：

- (1)  $a$  は、 $i_b$  を知っているとする。
- (2)  $a$  は、貸与された権限を利用できる回数や日時といった条件を、認可条件式  $R_{a \rightarrow b}$  として記述する。
- (3)  $a \rightarrow s : (i_b, R_{a \rightarrow b})$
- (4)  $s$  は  $F_s(b) = 1$  をセットする。

利用者  $b$  が  $a$  から貸与された権限を使ってサービス  $s$  を利用する手順：

- (5)  $b \rightarrow s : i_b$
- (6)  $s \rightarrow b : F_s(b) = 1$  かつ、 $\exists u \in U$  について  $R_{u \rightarrow b}$  が真ならサービスを提供する。そうでなければ提供しない。

2において、 $a$  から権限が貸与されていれば、 $R_{a \rightarrow b}$  が真となるため、サービスが提供される。

### 2.2 オフライン貸与利用プロトコル

オフライン貸与利用プロトコルは以下のように書くことができる。

利用者  $a$  から  $b$  への貸与手順：

- (1)  $a$  は、認可条件式  $R_{a \rightarrow b}$  を記述する。
- (2)  $a \rightarrow b : (i_a, i_b, R_{a \rightarrow b})$

利用者  $b$  が  $a$  から貸与された権限を使ってサービス  $s$  を利用する手順：

- (3)  $b \rightarrow s : (i_a, i_b, R_{a \rightarrow b})$
- (4)  $s \rightarrow b : F_s(a) = 1$  かつ、 $R_{a \rightarrow b}$  が真ならサービスを提供する。そうでなければ提供しない。

なお、どちらのプロトコルにおいても、あらかじめサービス  $s$  を利用できる権限を持つ利用者  $a$  には、 $F_s(a) = 1, R_{a \rightarrow a}$  に真がセットされているとする。

### 2.3 既存のサービスへの適用における要求

ここでは、貸与利用を考慮していない既存のサービスを単純サービスと呼び、単純サービスにできるだけ手を加えることなく貸与利用機能を付加することを考える。

単純サービス  $s$  は、認可関数  $F_s$  を持ち、任意の利用者  $a \in U$  に対して以下のプロトコルしか用意していないものとする。

ただし、 $Y$  の持つ秘密情報  $v_Y$  と、 $X$  の知る  $Y$  に関する情報  $u_Y$  によって、 $X$  が  $Y$  を認証することを、

$$X \rightarrow Y : \text{Auth}[u_Y, v_Y]$$

のように書く。公開鍵を用いた場合は  $v_Y, u_Y$  がそれぞれ秘密鍵、公開鍵に対応し、共通鍵を用いた場合は、 $v_Y = u_Y$  となる共通鍵と考える。パスワードを用いた場合も、 $v_Y = u_Y$  となるパスワードと考えることができる。

また、 $X$  と  $Y$  が鍵交換プロトコルを用いてワンタイム性を持つ鍵を共有し  $Z$  を暗号通信することを

$$X \rightarrow Y : \text{Code}[Z]$$

のように書く。

単純サービスの機能：

- (1)  $s \rightarrow a : \text{Auth}[u_a, v_a]$
- (2)  $a \rightarrow s : \text{Code}[i_a]$
- (3)  $s \rightarrow a : F_s(a) = 1$  ならサービスを提供する。そうでなければ提供しない。

ここでの認証や暗号通信の機能は、Web における SSL (Secure Socket Layer) を初めとする PKI (Public Key Infrastructure) において現実に普及している。

このような単純サービスに手を加えずに貸与利用機能を付加する方法として、利用者とサービスの間立ち、サービス側の貸与利用機能を代行するプロキシを導入することが考えられる。ただし、以下の点で注意

が必要である。

- 一般のサービスにおいて、利用者集合  $U$  は、利用者の追加や削除により頻繁に更新されることが考えられる。

すなわち、上記のプロトコルしか公開しない既存サービスに対して、プロキシは最新の  $U$  を把握することはできないということである。

### 3. プロキシを用いた貸与利用プロトコル

本節では、前述のプロトコルをプロキシを用いて単純サービスに適用する方法を提案し、その安全性を検証する。まずはじめに簡単のために安全性を考慮しないプロキシを用いたプロトコルを提案し、次に安全性を考慮したものへと拡張する。

#### 3.1 プロキシを用いたオフライン貸与利用プロトコル

前述の要求に従い、単純サービス  $s$  とプロキシ  $p$  を用いて貸与利用を実現する方法を、まずは安全性を考慮せずに提案する。以下では、オフライン貸与利用プロトコルのみを扱う。

以下で、下線部が新たに追加される部分である。

プロキシを導入した貸与手順：

- (1)  $a$  は、認可条件式  $R_{a \rightarrow b}$  を記述する。
- (2)  $a \rightarrow b$ :  $(i_a, i_b, R_{a \rightarrow b})$

プロキシを導入した利用手順：

- (3)  $b \rightarrow p$ :  $(i_a, i_b, R_{a \rightarrow b})$
- (4)  $p \rightarrow s$ :  $R_{a \rightarrow b}$  が真なら、 $i_a$
- (5)  $s \rightarrow p$ :  $F_s(a) = 1$  ならサービスを提供する。そうでなければ提供しない。
- (6)  $p \rightarrow b$ : 仲介したサービス

このようにして、単純サービスに貸与利用機能を付加することが可能である。ここでプロキシ  $p$  は、利用者集合  $U$  の情報を利用していないことを記す。これにより、利用者集合  $U$  に変更があっても、問題なく運用できることが分かる。

また、オフライン貸与利用プロトコルの場合は、プロキシ  $p$  は貸与時には何もなくて良いため、 $R_{a \rightarrow b}$  内の変数に永続性が必要な物がなければ、プロキシ  $p$  そのものがデータベースのような永続的な記憶領域を持つ必要がないことも特徴である。一方オンライン貸与利用プロトコルの場合は、貸与から利用までの間、 $R_{a \rightarrow b}$  を記憶しておかなければならない。

#### 3.2 想定される攻撃

以下では、利用者  $a$  から利用者  $b$  へ権利を貸与し利用する場合に想定される攻撃をあげる。ただし、中間者攻撃による通信路の乗っ取りは考えないものとする。

貸与者への成りすまし 被貸与者  $b$ 、プロキシ  $p$ 、またはサービス  $s$  が貸与者  $a$  に成りすまして  $a$  の権利を不正に利用または貸与する。

被貸与者への成りすまし  $a, p$ 、または  $s$  が被貸与者  $b$  に成りすまして、貸与された権利を不正に利用または貸与する。

サービスへの成りすまし  $a, b$ 、または  $p$  がサービス  $s$  に成りすまして、ほかの者の秘密情報を得る。

プロキシへの成りすまし  $a, b$ 、または  $s$  がプロキシ  $s$  に成りすまして、仲介する権利を不正に利用または貸与したり、ほかの者の秘密情報を得る。

貸与者の否認  $a$  が  $b$  に権利を貸与したことを否認する

被貸与者の否認  $b$  が  $a$  から権利を受け取って利用したことを否認する。

被貸与者の改ざん  $b$  が与えられた認可条件式  $R_{a \rightarrow b}$  を改ざん、または勝手に生成する

プロキシの改ざん プロキシ  $p$  が認可条件式  $(i_a, i_b, R_{a \rightarrow b})$  を改ざん、または勝手に生成する

プロキシによる再利用  $p$  が認可条件式  $(i_a, i_b, R_{a \rightarrow b})$  を再利用してサービスを受ける

第3者による再利用 第3者が認可条件式  $(i_a, i_b, R_{a \rightarrow b})$  を盗聴、再利用してサービスを受ける。

なお、網羅的に考えれば以下のような攻撃も考えられるが、権利貸与機能の性質から、これらは考えない。その理由も以下に述べる。

サービス、プロキシ、第3者の否認 権利を貸与する主体とされる対象は貸与者  $a$  か被貸与者  $b$  のみであり、それ以外の者には権利貸与の否認は無意味である。権利貸与以外の否認はここでは考えない。

貸与者、サービスの改ざん  $a$  または  $s$  が通信内容の改ざんを行ったとしても、もともとの単純サービスにおいて得られる以上の利益を得ることはないため、無意味である。

第3者の改ざん 第3者は権利の貸与、利用に介在しないため、中間者攻撃を考えなければ、改ざんは無意味である。

貸与者、サービスによる再利用 改ざんと同様、 $a$  または  $s$  が通信で得られる情報を再利用したとしても、もともとの単純サービスにおいて得られる以上の利益を得ることはないため、無意味である。

被貸与者の再利用 被貸与者  $b$  が通信内容を再利用する攻撃は、認可条件式  $R_{a \rightarrow b}$  の以下のような管理により、再利用する者が被貸与者である限りは、

無意味, または防がれているものと仮定する.

- 時間帯を表す認可条件式の場合: その時間帯外ではプロキシにより権利は無効であり, その時間帯内でもともと権利を何度も可能なので, 無意味である.
- 回数を表す認可条件式の場合: 認可条件式に ID が付与され, プロキシにおいて使用回数が記憶, 管理されるため, 回数を超えた権利の利用ははもとも無効であり, 無意味である.

### 3.3 安全性を考慮した貸与利用プロトコル

ここでは, 安全性を考慮した貸与利用プロトコルを述べる. 公開鍵を用いたプロトコルを中心に述べる. 共通鍵のみを用いる場合も, それぞれ  $u_a = v_a, u_b = v_b, u_p = v_p$  とおけば同様である. ただし次の節に述べるように, 安全性の点で違いが出てくる.

前提として, 利用者またはプロキシである  $X$  には秘密鍵  $v_X$  と公開鍵  $u_X$  が与えられる. 公開鍵  $u_X$  は広く公開されているとする.

秘密鍵  $v_X$  によるメッセージ  $m$  へのデジタル署名を

$$\text{sign}_{v_X}(m)$$

で表す. また, 公開鍵  $u_X$  による署名  $m'$  の復号を,

$$\text{sign}_{u_X}^{-1}(m')$$

であらわす.

安全性を考慮した貸与手順:

(1)  $a$  は, 認可条件式  $R_{a \rightarrow b}$  を記述する.

(2)  $a \rightarrow b: \text{Auth}[u_b, v_b]$

(3)  $a \rightarrow b: (i_a, i_b, R_{a \rightarrow b}),$

$$T_a := \text{sign}_{v_a}((i_a, i_b, R_{a \rightarrow b}),$$

$$K_a := \text{sign}_{u_p}(v_a)$$

安全性を考慮した利用手順:

(4)  $b \rightarrow p: \text{Auth}[u_p, v_p]$

(5)  $b \rightarrow p: (i_a, i_b, R_{a \rightarrow b}),$

$$T_b := \text{sign}_{v_b}(T_a, K_a)$$

(6)  $p: \text{sign}_{u_a}^{-1}(\text{sign}_{u_b}^{-1}(T_b)) = (i_a, i_b, R_{a \rightarrow b})$  か検証,

$$v_a = \text{sign}_{u_p}^{-1}(K_a) \text{ を得る}$$

(7)  $s \rightarrow p: \text{Auth}[u_a, v_a]$

(8)  $p \rightarrow s: R_{a \rightarrow b}$  が真なら,  $\text{Code}[i_a]$

(9)  $s \rightarrow p: F_s(a) = 1$  ならサービスを提供する. そうでなければ提供しない.

(10)  $p \rightarrow b: \text{仲介したサービス}$

### 3.4 検 証

前節で述べたプロトコルの安全性を検証する.

貸与者への成りすまし

被貸与者  $b$  もサービス  $s$  も第 3 者も, 貸与者  $a$  の

秘密鍵  $v_a$  を知らないため, プロトコルの 5 において  $T_a$  を生成することができず成りすませない.

ただし, プロキシ  $p$  は, プロトコルの 6 において  $v_a$  を得るため, 貸与者に成りすますことができる.

被貸与者への成りすまし

$a$  が被貸与者  $b$  に成りすまそうとしても  $b$  の秘密鍵  $v_b$  を知らないため, プロトコルの 5 において  $T_b$  を生成することができず成りすませない.

また  $s$  も  $p$  も第 3 者も,  $b$  の秘密鍵  $v_b$  を知らないため, プロトコルの 2 において被貸与者に成りすますことはできない.

サービスへの成りすまし

$a, b$ , または  $p$  がサービス  $s$  に成りすましたとしても, 得られる情報はプロトコルの 8 における  $i_a$  のみなので, 安全である.

プロキシへの成りすまし

$b$  も  $s$  も第 3 者も, プロキシ  $p$  の秘密鍵  $v_p$  を知らないため, プロトコルの 4 においてプロキシに成りすますことはできない. また秘密鍵  $v_a$  も知らないため, プロトコルの 7 においてもプロキシに成りすますことはできない.

また  $a$  は  $v_a$  を知っているため, プロトコルの 7 において成りすますことができる. しかし貸与者がプロキシに成りすまして, もともとの単純サービスにおいて得られる以上の利益を得ることはないため無意味である.

貸与者の否認

3 において,  $T_a$  は  $v_a$  を持つ者しか作ることができない.  $v_a$  はプロトコルの 6 において  $p$  も得ることができるので, 否認が可能になってしまう.

ただしこれは,  $p$  が成りすましや再利用, 改ざんを行ったときに限るので,  $p$  の安全性に依存する.

被貸与者の否認

5 において,  $T_b$  は  $v_b$  を持つ被貸与者  $b$  しか作ることができないため, 否認を防ぐことができる.

被貸与者の改ざん

$b$  が改ざんをすると, 3 における  $T_a$  が無効になるため安全である.

プロキシの改ざん

プロキシ  $p$  は,  $v_a$  をプロトコルの 6 において得ることができるので,  $T_a$  を生成または改ざんすることができてしまう.

プロキシによる再利用

プロキシ  $p$  は,  $v_a$  をプロトコルの 6 において得ることができるので, 仲介した権利を再利用することができてしまう.

### 第3者の再利用

盗聴が考えられる通信が発生する 3,5,8 について, 3,5 については, 再利用しようと思っても, 2 において,  $v_b$  を持つ  $b$  でなければ認証に失敗する.

8 においてはワンタイム性を持つ鍵による暗号通信を行っているので, 鍵を知るもの以外による再利用は無効である.

表 1, 2 に, 上記の安全性の検証を整理する.

## 4. 安全なプロキシの実現

前節において, 公開鍵を用いた場合でも, プロキシの貸与者への成りすまし, 改ざん, 再利用について安全でないこと, またそれに依存して貸与者の否認もできてしまうことが示された.

つまり, 提案するプロトコルにおいては, プロキシは信頼されているという前提が必要ということになる. 本節では, 信頼できるプロキシを実現する方法について, プロキシの分類毎に検討する.

### 4.1 独立した領域にプロキシを配置する場合

まず, 利用者やサービスとは独立した領域にプロキシを配置することを考える.

この場合は, サービスも利用者もプロキシを容易には信頼できないことが考えられる. その時には, 前節で述べた, プロキシの貸与者への成りすまし, 改ざん, 再利用について安全性に問題が残る.

### 4.2 サービスの領域にプロキシを配置する場合

次に, サービスの領域にプロキシを配置することを考える. 例えば, サービスの管理するサーバ用ネットワークに配置する場合などである.

この場合, サービスはプロキシを容易に信頼できることが考えられる. つまり, サービスとプロキシを同一視することができ, 前節の議論により, サービスが成りすまし, 改ざん, 再利用しても, 貸与者に否認をさせても無意味であるので, 全体として安全であるといえる.

ただし, この場合はサービスが自分の領域にプロキシを置かなければならないため, サービスが何もしない場合にはこの方法をとることができないことが問題である. サービスが対応しなくても, 利用者間で安全に権利貸与と機能を利用したいことがあることも考えられる.

### 4.3 利用者の領域にプロキシを配置する場合

最後に, 利用者, 特に被貸与者の領域にプロキシを配置することを考える. 例えば, 被貸与者の持つコンピュータに配置する場合などである.

この場合, 被貸与者はプロキシを容易に信頼できる

ことが考えられる. つまり, 被貸与者とプロキシを同一視することができる. すると, 前節の議論より, 被貸与者が貸与者の秘密鍵を知ることができることになる. これは,

- 被貸与者が貸与者に成りすましたり,
- 被貸与者が貸与者からの情報を改ざんする
- 貸与者が否認を成功させてしまう

ことにつながり問題が残る.

そこで以下では, 被貸与者の領域にプロキシをより安全に配置する方法を検討する.

利用者の領域におけるプロキシの安全性向上

ここでは, 被貸与者の領域に被貸与者自身がアクセスできない領域を作り, 安全性を向上させる方法を提案する.

- (1) まず, 被貸与者の領域には, 被貸与者自身がアクセスできないと同時に他の者が扱うことのできる領域を作ることができるものとする. 例えば, IC カードの耐タンパー領域に, IC カードベンダーが書き込むソフトウェアや, マルチユーザ OS において別の利用者アカウントにインストールするソフトウェア, 携帯電話においてダウンロードされた Java アプレットなどが考えられる.
- (2) 被貸与者は, この領域にプロキシを配置させる. そして, プロキシが外部と通信する際には, その通信の相手とタイミングと容量を監視する.
- (3) 監視される通信が, 被貸与者の権利の利用と異なる相手やタイミング, または容量である場合は, 通信を遮断する.

これにより, 監視される通信が暗号通信であっても, プロキシが不正な行動を行っている場合には発見し食い止めることができる.

ただし, 被貸与者とプロキシが結託した場合には, 上記の被貸与者とプロキシを同一視する場合と同様に被貸与者の安全性に問題が残る. ここで述べた方法は, プロキシの信頼を被貸与者に委託したものであるといえる. 現実の応用において, 貸与者と被貸与者間でのみ信頼でき, 被貸与者とプロキシが実行環境などの制約により結託できないことが明らかである場合には有用である.

## 5. 関連研究

公開鍵を利用して権利の貸与を行う既存研究として Proxy Signature<sup>1)</sup> がある. Proxy Signature は署名者が代理人に対してデジタル署名を生成する権利を貸与もしくは譲渡する技術である. この Proxy Signa-

表 1 公開鍵を用いた場合の成りすましへの安全性

	貸与者への成りすまし	被貸与者への成りすまし	サービスへの成りすまし	プロキシへの成りすまし
貸与者の成りすまし	-	安全 ( $T_b$ を作れない)	安全 ( $i_a$ しか得ることができない)	- (メリットがない)
被貸与者の成りすまし	安全 (貸与者の秘密鍵を知らない)	-	安全 ( $i_a$ しか得ることができない)	安全 (プロキシの秘密鍵を知らない)
サービスの成りすまし	安全 (貸与者の秘密鍵を知らない)	安全 (被貸与者の秘密鍵を知らない)	-	安全 (プロキシの秘密鍵を知らない)
プロキシの成りすまし	安全でない (貸与者の秘密鍵を知りえる)	安全 (被貸与者の秘密鍵を知らない)	安全 ( $i_a$ しか得ることができない)	-
第 3 者の成りすまし	安全 (貸与者の秘密鍵を知らない)	安全 (被貸与者の秘密鍵を知らない)	安全 ( $i_a$ しか得ることができない)	安全 (プロキシの秘密鍵を知らない)

表 2 成りすまし以外の安全性

主体 \ 行為	否認	改ざん	再利用
貸与者	安全でない (プロキシの安全性に依存)	-	-
被貸与者	安全 ( $T_a$ が証明となる)	安全 ( $T_a$ を作れない)	- (認可条件式をプロキシが管理)
サービス	-	-	-
プロキシ	-	安全でない (貸与者の秘密鍵を知りえる)	安全でない (貸与者の秘密鍵を知りえる)
第 3 者	-	- (中間者攻撃は考えない)	安全 (通信の暗号化と貸与内容への署名)

ture のアイデアを、電子鍵システムでの鍵の譲渡および複製に応用した研究が文献<sup>4),5)</sup>である。この研究では鍵の又貸しについても実現方法を提案している。

プロキシを導入して認証を代理させる既存の技術として Single Sign On(SSO)<sup>2),3)</sup>がある。SSOは、一度の認証でマルチサービスへのアクセスを可能にする技術である。文献<sup>2)</sup>によると、プロキシに対する信頼の状態によって以下のような分類がされている。

- Local Pseudo-SSO  
利用者のマシン内にプロキシが存在する
- Proxy-Based Pseudo-SSO  
独立したプロキシサーバが存在する
- Local True SSO  
利用者に信頼されたプロキシが利用者のマシン内に存在する
- Proxy-Based True SSO  
サービスと利用者に信頼された独立したプロキシが存在する

サービス側を変更せずに複数の権利を管理する先行研究も存在している<sup>3)</sup>。しかしながら SSO では一般的に他の利用者の権利を利用者が貸与により得ることは

考えられていない。このため、本論文のように各分類における貸与利用における他人の権利の安全性については考慮されていない。

## 6. おわりに

本論文では、利用者と既存システムの間で権利利用時にのみ代理処理をするプロキシを導入させることにより、既存システムに変更を加えることなく貸与利用機能を付加させる方法を提案した。プロキシを用いる場合は一般にプロキシの信頼性が問題となるが、提案プロトコルにおける問題の影響を明らかにし、プロキシの信頼性を実現するための方法をプロキシの分類毎に検討し、特に利用者の領域にプロキシを置いた場合に、プロキシの信頼性を被貸与者に委託する方法を提案した。

今後は実装による評価とともに、最後に述べたようなプロキシの信頼性を被貸与者に委託する場合に、複数の利用者間での委託を検討することが今後の課題である。

## 謝 辞

本論文は、科学研究費補助金・若手研究 A ( 課題番号 : 18680009 ) によるものである。議論いただいた研究室の諸氏に感謝する。

## 参 考 文 献

- 1) M.Mambo, K.Usuda, E.Okamoto, “Proxy signatures: Delegation of the power to sign messages”, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, E79-A(9):1338-1354, 1996.
  - 2) Andreas Pashalidis, ChrisJ. Mitchell, “ A Taxonomy of Single Sign-On Systems”, Lecture Notes in Computer Science, Vol.2727, pp.249-264, Springer Berlin/Heidelberg, 2003.
  - 3) 野林大起, 中村豊, 池永全志, “ハードウェアトークンと鍵管理サーバを用いたシングルサインオンシステムの開発”, 電子情報通信学会技術研究報告インターネットアーキテクチャ, Vol.106, p.7, 2006.
  - 4) 山崎知美, 中村徹, 馬場謙介, 安浦寛人, “ 局所的に複製と譲渡が可能な権利管理手法”, 2007 年暗号と情報セキュリティシンポジウム (SCIS 2007), 2007.
  - 5) Tomomi Yamasaki, Toru Nakamura, Kensuke Baba, and Hiroto Yasuura, “A Door Access Control System with Mobile Phones”, Proc. Personal Wireless Communications (PWC'07), Lecture Notes in Computer Science, to appear.
-