# 分散環境における，プロクシを利用した柔軟なセキュリティ制御

双紙正和† 加藤丈治† 前川守†

我々は現在，分散環境におけるセキュリティアーキテクチャ，*Saga Security System*の研究開発を行っている．本論文では，Saga Security Systemの1つの応用例，**プロクシ**について議論する．プロクシのアプローチの有用性は早くから知られ，さまざまなシステムが「プロクシ」を実現してきた．我々は，このようなプロクシの有用性をもたらすその本質的な特徴は，**間接化・独立性・集中化**の3点であると考える．そして，そのような特徴を分散環境におけるセキュリティに応用したとき，非常に柔軟なセキュリティ制御を提供することが可能となる．このようなプロクシによるセキュリティ制御のうち，本論文は，信頼関係の制御，ネットワークワイド・カプセル化，クライアント・プライバシ，任意・強制アクセス制御，ロールによるアクセス制御，セキュリティ管理の6点について議論する．残念ながら，従来の認可モデルにおいては，プロクシによる柔軟なセキュリティ制御を十分に扱うことはできない．ここで，Saga Security Systemは，分散環境における柔軟で高度なセキュリティ制御を行うために，**サービスパス**に基づいた**Saga認可モデル**を提供する．このSaga認可モデルであれば，プロクシについても応用例の1つとして統一的に扱うことができる．

# Flexible Protection Realized by Proxies in Distributed Systems

MASAKAZU SOSHI,† TAKEHARU KATO† and MAMORU MAEKAWA†

We are designing and developing *Saga Security System*, a security architecture in distributed systems. In this paper, we discuss flexible protection provided by *proxies* realized in Saga Security System. Now, although the usefulness of 'proxies' is well-known and various systems have so far supported them in various fashions, we argue that the essential properties of such proxies can be classified into only three distinct ones: *indirection*, *independence*, and *centralization*. With respect to security, the three properties of such proxies are also advantageous in that proxies can provide highly flexible protection by utilizing the properties. Among advantages in regard to security control by proxies and discussed in this paper are: *control over trust relationships*, *network-wide encapsulation*, *client privacy*, *discretionary/mandatory access control*, *role-based access control*, and *security management*. Unfortunately, traditional authorization models are not fully applicable to flexible protection offered by proxies. However, the authorization model in Saga Security System (*Saga authorization model*) supports the novel concept of a *service path* and is able to deal with protection by proxies in uniform way.

## 1. はじめに

近年一般的になりつつある分散コンピューティング環境では，広範な地域におけるコンピュータ資源の共有が可能になる一方で，セキュリティの確保が非常に困難である．しかし，従来のセキュリティ技術は集中型システムを想定したものが多く，新しい環境の変化に十分に対応できていない[1]．

そこで我々は，そのような環境におけるセキュリティアーキテクチャ，*Saga Security System*の研究開発を行っている[2),3)]．本論文は，Saga Security Systemの1つの応用例，**プロクシ**による柔軟なセキュリティ制御に重点をおいて議論するものである．

プロクシのアプローチの有用性は早くから知られ，さまざまなシステムが「プロクシ」を実現してきた☆．ところが，従来のシステムが実現してきた「プロクシ」の内容はさまざまであり，プロクシの持つ本質的な性質や利点を理解しにくくなっている．そのために，現在の「プロクシ」のアプローチは経験的なものにとどまることが多く，その有用性を十分に理解したり生かしたりすることが困難である．

そこで我々は，プロクシの本質をとらえた一般的な

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, University of Electro-Communications

☆ 本研究で提案するプロクシと区別するために，以下では，従来研究やシステムにおけるものは「プロクシ」と記述する．

フレームワークをまず最初に定義し，プロクシの特徴や利点について議論する．次に，そのようなプロクシをセキュリティへ応用したとき，柔軟で高度なセキュリティ制御を提供できることを示す．ところが，このような柔軟なセキュリティ制御は従来の認可モデルでは十分に扱うことができない．そこで，Saga Security System の 1 つの応用例として，そのようなセキュリティ制御を可能とするプロクシの実現を提案する．

本論文の残りの部分は以下のように構成される．まず，2 章において従来の「プロクシ」について議論する．次に，3 章でプロクシの一般的な側面について議論を行い，特にセキュリティ制御におけるプロクシの利点を明らかにする．そして，4 章において Saga Security System の認可モデルについて議論し，その応用例を 5 章で考える．そして，6 章において考察を加え，最後に 7 章において結論を述べる．

## 2. プロクシに関する従来の研究

従来の研究においては,「プロクシ」という用語はさまざまな意味で用いられてきた．たとえば，Shapiro[4] は，分散環境において，あるサービスを提供するプロセス群に「プロクシ」を用いた構造的な構成を持たせることにより，カプセル化などのさまざまな優れた特徴を持たせることができることを示した．この場合の「プロクシ」は，機構面から見ると，遠隔手続き呼び出しにおけるスタブに近い．また，World Wide Web における「プロクシ」サーバ[5]は，クライアントからの HTTP によるリクエストを目的サーバに対する適当なプロトコルに変換する．双方向の通信における中継を可能にする DeleGate[6] なども「プロクシ」の 1 種と見なすことができる．さらに，オブジェクト指向システムにおける強制アクセス制御を実現するために，message filtering というアプローチが提案されている[7]が，これを実現する message filter はまさに「プロクシ」である．また，認証機構においては従来から「プロクシ」は重要な役割を果たしてきた[8]．

このように「プロクシ」はさまざまに実現されており，プロクシの有用性は広く認識されているといえる．しかしながらその一方で，プロクシの持つ本質的な性質や利点は理解しにくくなっている．したがって，現在の「プロクシ」のアプローチは経験的なものにとどまることが多く，その有用性を十分に理解したり生かしたりすることは困難である．

## 3. 分散環境におけるプロクシ

2 章で議論してきたように，従来の「プロクシ」の内容は非常に多岐にわたる．そこで，プロクシについて議論する前に，プロクシの本質的なフレームワークについて定義する必要がある．このために，2 章の議論をもとにして，我々はプロクシを

> あるリクエストを受けたとき，与えられたルールに基づいて，新たなリクエストを送り出すエージェント

のように定義する．この定義は，少なくともその一部を従来の「プロクシ」に適用できるように，意図的にかなり緩やかなものになっている．この章では，このようなプロクシが持つ特徴や利点について議論する．

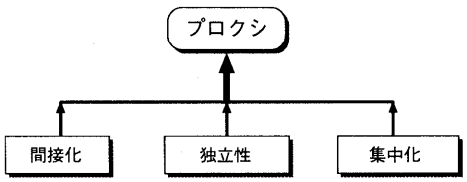


図 1 プロクシの概念
Fig. 1 The concept of proxy.

### 3.1 プロクシの本質

今まで議論してきたように，従来の「プロクシ」のアプローチはアドホックなものが多く，プロクシの有用性をもたらす，その本質的な特徴について考察されることはほとんどなかった．

そこで，本論文では，プロクシの本質をまず明らかにする．我々は，プロクシの本質とは，次の 3 点に抽象化できると考える（**図 1** 参照）：

- **間接化**
- **独立性**
- **集中化**

以下の項で，それぞれについてより深く議論する．

#### 3.1.1 間 接 化

プロクシを利用した分散コンピューティングでは，クライアントがあるオペレーションを要求するときは，直接サーバにリクエストを出さずに，まずプロクシにリクエストを行う．そして，このプロクシが，定められたルールに基づいてサーバに適切なリクエストを出す．このように，プロクシは分散環境において**間接化**を提供するものである．

このような間接化は，以下のような重要な利点を提供する：

［**制御**］クライアントとサーバとの 2 つの実体，あるいは 2 つのドメイン（ネットワークや組織）の間におけるインタラクションを，プロクシが明示的に制御することができる．たとえば，プロクシによる間接化は，プロトコル変換，filtering に適

している．

［関係の表現］プロクシによって，クライアントとサーバとの間におけるさまざまな関係を実現・維持することができる．たとえば，プロクシはオブジェクト指向システムにおけるリレーション[9]をより一般化し，分散環境において実現することができる．

前者の分類は，クライアントとサーバにおける関係を動的・明示的に変更を加えるもの，後者の分類は，両者の間に存在する比較的静的な関係を表現・維持するものと考えられる．

#### 3.1.2 独立性

プロクシは，エージェントであり独立した実体である．そこで，分散環境においてプロクシは，負荷分散のために，あるいは障害を避けるために移動することができる．また，他のプロクシと協調して，階層的グループのような柔軟な構成を持つことが可能である．

このように，プロクシの持つ**独立性**という性質によって，機能や性能のトレードオフを考慮した柔軟性を提供することができ，分散システムの利点を享受することができる．

#### 3.1.3 集中化

プロクシは，複数のリクエストを受けて複数の新たなリクエストを送出することができるため，ハブ（hub）の機能を提供するのに適している．つまり，分散環境において，処理の**集中化**が可能となる．

分散システムは，集中型システムと比較すると，より複雑な管理が必要となる．このような状況は，集中化を実現できるプロクシによって，分散システムと集中型システムとの利点を統合することによって改善される．このために，グループ，ドメインごとにデータ処理，管理などを集中的に行うプロクシを用意する．そして，それらの情報の管理や制御をそのプロクシが司るようにすればよい．

### 3.2 セキュリティ制御におけるプロクシの利点

3.1 節の議論より，分散環境におけるプロクシの有用性が明らかになった．

本論文は，プロクシによるセキュリティ制御に重点をおいて議論する．プロクシは，間接化・独立性・集中化という特徴を持つことによって，以下のような重要なセキュリティ上の利点を持つことになる（**図2** 参照）：

- **信頼関係の制御**
- **ネットワークワイド・カプセル化**
- **クライアント・プライバシ**
- **任意・強制アクセス制御**

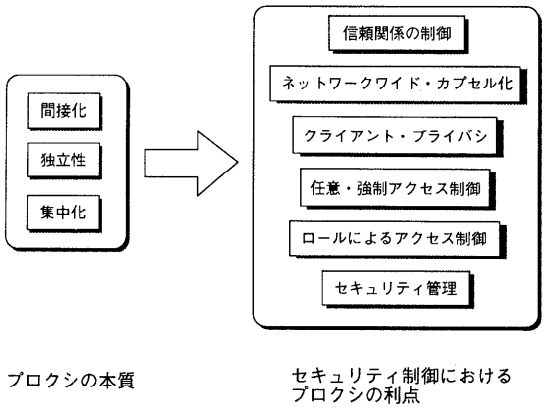


**図2** セキュリティ制御におけるプロクシ
Fig. 2 Proxies in security control.

- **ロールによるアクセス制御**
- **セキュリティ管理**

これらについて，以下で議論する．

#### 3.2.1 信頼関係の制御

実体 $A$ が実体 $B$ を信頼しているというのは，$B$ がある役割を演じるということを，$A$ が期待しているということと定義できる[10]．単一でグローバルな信頼できる実体が存在しえない分散システムにおいては，このような**信頼関係**の確立・制御は重要な問題となる．さらに，実際のシステムでは，信頼関係の度合に応じて一部の権限だけエージェントに委譲して処理を代行させたいといった，複雑な信頼関係の制御が必要となることが多い[8]．

3.1 節で議論したような，プロクシの間接化・独立性という性質は，このような信頼関係の確立・制御に適している．たとえば，上記の実体 $A$ と $B$ との間に信頼できるプロクシを介在させて，さまざまな度合の信頼関係を確立・制御することが可能である．

#### 3.2.2 ネットワークワイド・カプセル化

クライアントがプロクシに対してあるサービスをリクエストするとき，クライアントが見ることができるのはそのプロクシだけである．つまり，そのプロクシがさらに他のエージェントを呼び出すことがあったとしても，クライアントはその詳細について知ることはできない．

このような，集中型システムにおけるカプセル化を，分散システムにおいて協調するプロセス群について拡張した**ネットワークワイド・カプセル化**を実現するのに，プロクシは適している[4]．

#### 3.2.3 クライアント・プライバシ

ある処理を行うときに，そのクライアントの情報すべてが必要になるわけではない．また，プライバシの

ためにクライアントが情報を隠したいこともある．プロクシの間接化によって，これが可能になる．

3.2.2 項で述べたカプセル化は，プロクシがサーバをクライアントから隠すものであった．これは，逆に考えることもできる．すなわち，プロクシは，クライアントに関する情報をサーバに対して隠しているのである．たとえば，サーバは，自分が受けとったリクエストがだれによるものなのかを知ることができない．つまり，分散環境におけるセキュリティ制御で，プロクシは自然にプライバシを提供するアプローチといえる．

#### 3.2.4 任意・強制アクセス制御[11)]

集中型システムにおいてアクセス制御を行うのは，リファレンスモニタの機能を実現するセキュリティカーネルやオペレーティングシステム[12)]である．プロクシは間接化・独立性・集中化という性質を持ち，分散環境においてアクセス制御を実現するのに非常に都合が良い．これを端的に表す一例として，2 章で述べた message filter があげられる．

#### 3.2.5 ロールによるアクセス制御

従来の任意・強制アクセス制御の枠組み[11)]は，現在の分散環境や，企業組織におけるセキュリティポリシーに十分に対応できない．このような状況を打破するために，近年，**ロールによるアクセス制御**（role-based access control，RBAC）[13)]の研究がさかんである☆．

RBAC における**ロール**（role）とは，ある 1 つの仕事を遂行するのに必要なるオペレーションの集合である．ユーザはオペレーションに対する権限を直接与えられるのではなく，ロールを割り当てられることによって，そのロールが持つオペレーションを実行できるようになる．このようにロールを導入することによって，システム全体の権限の管理は大幅に単純化され，また柔軟なアクセス制御を行うことが可能になる．

プロクシによって，RBAC を分散環境において自然に実現することができる．そのためには，ロールとプロクシとを 1 対 1 対応させればよい．そして，ユーザがロールにおける仕事を遂行しようとするときは，プロクシに対してリクエストを行う．プロクシは，リクエストしたユーザがそのロールを割り当てられているかどうか判断する．そして，許可される場合，ロールに応じてそのオペレーションを実行したり，さらに他のエージェントに対してリクエストしたりすればよい．

#### 3.2.6 セキュリティ管理

3.1.3 項で議論したとおり，分散システムは，集中型システムよりも管理が困難である．

セキュリティの性質上，そのような管理の困難さが問題になることがある．たとえば，あるユーザの権限を revoke しようとしたとき，そのセキュリティ情報のアップデートに時間がかかったり矛盾が生じたりすれば，revocation の意味がなくなってしまう．さらに，組織全体のセキュリティポリシーを一括して管理・変更することも困難になる．

以上のような状況は，3.1.2 項で議論したように，組織などの単位でプロクシを用意し，セキュリティ管理を集中的に行うことによって救済される．さらに，組織の階層的構成や，性能や管理の手間などのトレードオフを考慮して，複数のプロクシが協調処理を行うようにすることもできる．

## 4. Saga 認可モデルの概要

今まで議論してきたように，プロクシによって，分散環境において柔軟なセキュリティ制御を提供することができる．ところが，このような制御に適したセキュリティモデルや機構はほとんどない．

現在我々は，分散環境におけるセキュリティアーキテクチャ，*Saga Security System* の研究開発を行っている．Saga Security System における認可モデル，**Saga 認可モデル**（*Saga authorization model*）は，分散環境における柔軟な認可モデルを提供することができ，プロクシについても応用例の 1 つとして統一的に扱うことができる．

この章では，Saga 認可モデルの概要について議論し，次に Saga 認可モデルにおける，プロクシによるセキュリティ制御について述べる．Saga 認可モデルの詳細については，たとえば文献 2) を参照されたい．また，Saga Security System におけるエージェントは *Saga Agent* と呼ばれ，それが提供するオペレーションは**サービス**（service）と呼ばれる．

### 4.1 サービスコンテクスト

従来の認可モデルは read/write といったプリミティブな操作だけを対象にするため，オブジェクト指向システムにおけるカプセル化の概念に適していない．そこで，オブジェクト指向システムにおいて，プリミティブな操作に対してではなく，メソッドに認可を与えるようなモデルの研究がさかんである[14)]．

また，あるサービスについて，その Saga Agent がどのユーザの代理であるかということを Saga 認可モデルによって表現できれば，分散環境における信頼関係に応じたセキュリティ制御を行える．

以上の議論をもとにして，Saga 認可モデルでは以

☆ RBAC は，厳密にいえば任意・強制アクセス制御には分類できない[13)]．そこで，3.2.4 項と独立した 1 項を設けた．

下のようなモデル化が行われる．あるSaga Agentを起動したユーザを，そのSaga Agentの**実行時ユーザ**（current user）と定義する．本論文ではSaga Agentを $o$ で表し，さらに，Saga Agent $o$ が提供するサービスを $o.s$ のように表記する．そして，実行時ユーザとサービスとの組：

$(u, o.s)$

を**サービスコンテクスト**（service context）として定義する．本論文では，サービスコンテクストを $\sigma$ によって表記する．

Saga認可モデルと従来の認可モデルとの大きな違いは，Saga認可モデルは，このサービスコンテクストを基本的な認可の単位にしているという点である．つまり，サービスコンテクストが従来のセキュリティモデルのサブジェクト/オブジェクト[12]に相当するのである．このようにして，Saga認可モデルでは統一的で一様なモデル化がなされている．

### 4.2 分散環境におけるサービスパス

Saga認可モデルは，分散環境において柔軟なセキュリティ制御を行うために，サービスコンテクストをさらに発展させ，以下のようなモデル化を行っている．

まず，サービスコンテクストの起動列：

$\sigma_1\sigma_2 \ldots \sigma_i$

を，**サービスパス**（service path）と定義する．この論文では，サービスパスを $\alpha$ で表す．ここで，$\sigma_1\sigma_2 \ldots \sigma_i$ は，$\sigma_1$ が $\sigma_2$ を起動し，$\sigma_2$ がさらに $\sigma_3$ を起動し，以下同様にして，最後に $\sigma_{i-1}$ が $\sigma_i$ を起動する，といった状況を表しているものとする．

そして，Saga認可モデルでは，あるサービスコンテクストを起動する認可を，**実体に対してではなく，サービスパスに対して与えることができる**ようになっている．このために，サービスパス $\alpha$ とサービスコンテクスト $\sigma$ との2つ組：

$(\alpha, \sigma)$

を**リクエストペア**（request pair）と呼ぶことにし，それによって $\alpha$ が $\sigma$ の起動を要求したような状況を表すものとする．そして，Saga認可モデルでは，このリクエストペアに対して認可が与えられる．

より具体的には，$(\alpha, \sigma)$ $(= (\sigma_1\sigma_2 \ldots \sigma_i, \sigma))$ が認可されているということは，サービスコンテクストの起動列が $\sigma_1\sigma_2 \ldots \sigma_i$ に一致しているときに限り，$\sigma$ の起動が許可されるということを意味する．よってSaga認可モデルでは，分散環境において協調するエージェントに対して，ある定められた協調作業だけ許可する，といった高度なセキュリティ制御が可能になる[2]．

他の認可モデルには見当たらないこのようなアプローチが，Saga認可モデルが優れている点であり，これによって，4.1節で議論した点が統一的かつ発展的に統合されていることが分かる．

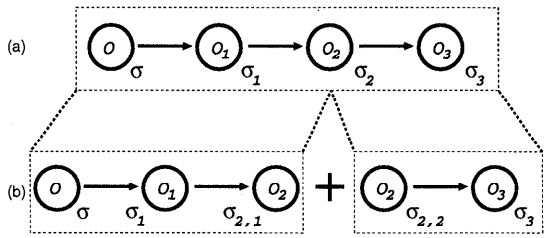


**図3** サービスパスの拡張
Fig. 3 Service path extension.

### 4.3 サービスパスの拡張

この節では，プロクシによる柔軟なセキュリティ制御を提供するために，Saga認可モデルの応用の1つである**サービスパスの拡張**（*service path extension*）について議論する．

#### 4.3.1 サービスパスの拡張とは

サービスパスの拡張について説明するために，リクエストペア $(\sigma\sigma_1\sigma_2, \sigma_3)$ を例として考える．すなわち，$\sigma$ が $\sigma_1$ を呼び出し，次に $\sigma_1$ が $\sigma_2$ を呼び出し，そのサービスパス $\sigma\sigma_1\sigma_2$ に基づいて，$\sigma_2$ が $\sigma_3$ を呼び出すような状況である（**図3**(a)）．

ここで，サービスコンテクスト $\sigma_2$ に対応する2つのサービスコンテクスト $\sigma_{2,1}$ および $\sigma_{2,1}$ を考え，さらに，2つのリクエストペア $(\sigma\sigma_1, \sigma_{2,1})$ と $(\sigma_{2,2}, \sigma_3)$ とで1つのリクエストペア $(\sigma\sigma_1\sigma_2, \sigma_3)$ と同等の処理を行うことを考える（図3(b)）．すなわち，$(\sigma\sigma_1, \sigma_{2,1})$ というリクエストが来たとき，それが認可されると，Saga Agent $o_2$ は $\sigma\sigma_1\sigma_{2,1}$ というサービスパスに対応するリクエストを行うことなく，**新たなリクエスト**として $(\sigma_{2,2}, \sigma_3)$ というリクエストを行うのである．Saga認可モデルにおける認可の基本は1つのサービスパスを持つリクエストペアであるが，複数のサービスパスを疑似的に1つのサービスパスと見なした認可の設定を考えることもできる☆．これがサービスパスの拡張である．

サービスパスの拡張は，プロクシの構築にきわめて適している．図3では，Saga Agent $o_2$ がプロクシに相当する．これについて，以下の項で議論する．

#### 4.3.2 サービスパスの拡張によるプロクシの実現

サービスパスの拡張によって，3.2節で述べたような，セキュリティ制御におけるプロクシのメリットを

☆ つまり，サービスパスの拡張は，Saga認可モデルの枠組みに完全に含まれている．

すべて実現することができる．ここでは，再び例として図3のような状況について考えるものとする．

#### 4.3.2.1 信頼関係の制御

図3の(a)と(b)との違いについて考えてみる．まず図3(b)では，プロクシ $o_2$ は，Saga Agent $o_1$ からリクエスト $(\sigma\sigma_1, \sigma_{2,1})$ を受け取り，そのリクエストに基づいて，$o_3$ に対して新たに $(\sigma_{2,2}, \sigma_3)$ というリクエストを行っている．そして，$o_3$ は，単に $o_2$ から出されたリクエスト $(\sigma_{2,2}, \sigma_3)$ についてのみセキュリティ制御を行う．これは，プロクシ $o_2$ がサービスパスの拡張を行ったかどうかを $o_3$ は関知しないという状況，すなわち，$o_3$ は $o_2$ を信頼しているという状況を表している．逆に，もし $o_3$ が $o_2$ を完全に信用したくない場合は，$o$ から $o_1$，$o_2$ に至るサービスパスについてセキュリティ制御すればよいだけである．つまり，$o_2$ が $\sigma\sigma_1\sigma_2$ という協調をしているときに限り，$\sigma_3$ の起動を許可するようにする☆．これが図3(a)の状況である．このように，サービスパスの拡張によって，信頼関係を制御することができる．

#### 4.3.2.2 ネットワークワイド・カプセル化

プロクシにおけるカプセル化については，一連のサービスコンテクスト（$\sigma$，$\sigma_1$，$\sigma_{2,1}$，$\sigma_{2,2}$，$\sigma_3$）において，$o_3$ および $\sigma_3$ が，$o$ および $o_1$ に対して隠されているということがこれに当てはまる．

#### 4.3.2.3 クライアント・プライバシ

図3におけるリクエストにおいては，プロクシ $o_2$ がサービスパスの拡張を行っているので，$o_3$ は自分が受けるリクエストがだれによるものなのかを知ることができない．

#### 4.3.2.4 任意・強制アクセス制御

通常，クライアントとサーバとの間におけるインタラクションをセキュリティ制御しようとするとき，サーバ側でのみ制御を行う．しかしながら，プロクシによる間接化を提供することによって，プロクシとサーバという，2つの段階におけるセキュリティ制御が可能になり（図3でいうと，プロクシ $o_2$ とサーバ $o_3$），非常に柔軟な制御を提供することができる．

#### 4.3.2.5 ロールによるアクセス制御

3.2.5項で議論したように，プロクシ $o_2$ をロールに対応させればよい．また，4.3.2.4項に述べたように，サービスパスの拡張によって，2段階のセキュリティ制御が可能になる．そこで，ロールの段階による制御だけが行われていた従来のRBACよりも，はるかに柔軟な制御が可能となる．

#### 4.3.2.6 セキュリティ管理

サービスパスの拡張によって，認可の管理を容易にすることが可能になる．再び図3について考えてみよう．通常，図3(a)のような状況では，Saga Agent $o_3$ は $(\sigma\sigma_1\sigma_2, \sigma_3)$ に対する認可を管理しなければならない．ところが図3(b)のような状況では，$o_3$ は $(\sigma_{2,2}, \sigma_3)$ だけを管理すればよく，サービスコンテクスト $\sigma$ および $\sigma_1$ について知る必要がなくなってくる☆☆．この効果は，サービスパスが長くなればなるほど，あるいはサービスパスの拡張が連続的に行われれば行われるほど大きくなってくる．

## 5. 応用：投票プロクシ

この章では，プロクシのアプローチの有効性，および，それをサービスパスの拡張によって効率良く実現できることを示すために，例として，投票システムを構築することを考える．ただし，ここではサービスパスの拡張の有効性を示すことに重点をおき，完全な投票システムを構築することは目指さない．そこで，たとえば暗号を高度に応用した投票プロトコルについては，文献15)などを参照されたい．また，ここでは表記を単純にするため，サービスコンテクストとサービスを同一視する．

では，この章で議論する投票システムについて説明しよう．まず，$n$ 人の投票者を $\mathsf{voter}_i$（ただし，$i = 1, \ldots, n$）とする．$\mathsf{voter}_i$ は，自分のエージェント（Saga Agent）$v_i$ のサービス voteReq を利用して投票を行う．voteReq はインタフェースにすぎず，投票するために実際はプロクシ（Saga Agent）$p$ の提供するサービス vote を呼び出す．$p$.vote は，リクエストを受け取ると，その投票者が投票できるかどうかを調べるために，Saga Agent $o_1$ のサービス checkVoter をリクエストする．$o_1$ は，投票者に関する情報のデータベースやログを利用して，投票者が資格を持っていることや複数回投票していないことを調べてその結果を $p$ に返す．もし，それらの条件が満たされている場合，$p$ はその投票者の代わりに投票を行う．このために，$p$.vote は Saga Agent $o_2$ のサービス voteOnBehalfOfVoter を起動する．

さて，ここでの，Saga認可モデルにおける投票シ

☆ これは，$o_2$ が $o_1$ を代行（間接的に $o$ をも代行）しているときに限り，$o_2$ が $o_1$（および $o_1$）に関する処理（ここでは $\sigma_2$ に相当）を行うことを許可されているような状況と見ることもできる．具体的な例として，旅行代理店が，旅行者（クライアント）の予約作業を代行しているような状況[8]を考えれば分かりやすいかもしれない．

☆☆ これは $o_3$ が $o_2$ を信頼していることによる．

ステムの条件は以下のとおりである：

[条件1] $p$ が，voter$_i$ による $v_i$ を代行している場合にのみ，$o_1$ が checkVoter を実行する．

[条件2] 投票者がだれに投票したかというプライバシを守るために，実際の投票は $p$ が代行する．

このような条件を満たす投票システムは，以下のように与えられる：

[ステップ1] voter$_i$ は，エージェント $v_i$ によって，リクエストペア

$(v_i.\text{voteReq}, p.\text{vote})$

を，プロクシ $p$ に対して送る．

[ステップ2] $p$ は，voter$_i$ が投票できるかどうかを確かめるために，$o_1$ に対して

$(v_i.\text{voteReq} \cdot p.\text{vote}, o_1.\text{checkVoter})$

というリクエストを行う．このとき，$(p.\text{vote}, o_1.\text{checkVoter})$ は許可されないものとする．

[ステップ3] $o_1$ は voter$_i$ が投票できるかどうかを調べ，結果を $p$ に返す．$o_1$ が許可した場合に限り，$p$ は voter$_i$ に代わって $o_2$ に対して投票

$(p.\text{vote}, o_2.\text{voteOnBehalfOfVoter})$

を行う．

3.2.1 節で議論したとおり，条件1については，ステップ2においてサービスパス $v_i.\text{voteReq} \cdot p.\text{vote}$ に基づいたリクエストがなされていることにより実現される．さらに条件2については，ステップ3においてサービスパスの拡張によるクライアント・プライバシがなされていることから満たされる．また，図4のシステムは，各クライアントごとに投票システムを実現するのではなく，プロクシを利用して構築されているため，3章で議論した利点をすべて持っていることになる．

ここで，Saga 認可モデルの汎用性と柔軟性に注意されたい．たとえば，上記のプロトコルにおいて，状況によっては，$p$ を完全に信頼できる実体と見なして，ステップ2においてサービスパスの拡張，つまり $(p.\text{vote}, o_1.\text{checkVoter})$ というリクエストを認可することも可能である．逆に，ステップ3において，$(v_i.\text{voteReq} \cdot p.\text{vote}, o_2.\text{voteOnBehalfOfVoter})$ というリクエストしか認可させないようにすれば，$p$ において必要とされる信頼性を減じることができる．このように，Saga 認可モデルは，さまざまな状況に柔軟に対応できる高度なセキュリティ制御を提供できる☆．

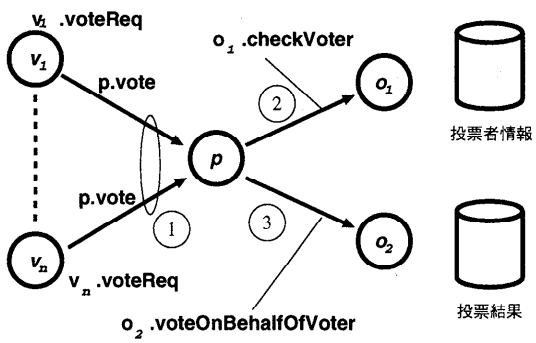


図4 投票プロクシ
Fig. 4 Voting proxies.

## 6. 議　　論

この章では，本論文で提案した，プロクシの概念およびSaga認可モデルにおけるプロクシの実現について考察する．

今まで議論してきたとおり，プロクシを利用することによって，分散環境においてきわめて柔軟で高度なセキュリティ制御を提供できる．その反面，セキュリティ制御を行うプロクシに関していえば，通常のエージェント以上にプロクシに対するセキュリティの確保が重要となる．たとえば，3.1節で議論したような，間接化・独立性・集中化というプロクシの本質は，逆にいえば，プロクシを実行している環境のセキュリティ的な弱点（vulnerability）になりかねない．したがって，プロクシの実行環境は非常に安全性が要求されるのみならず，理想的には，形式的な仕様記述によって安全なプロクシを実現する必要があろう．

## 7. 結　　論

我々は，現在，オープンな分散環境におけるセキュリティアーキテクチャ，*Saga Security System*の研究開発を行っている．本論文は，Saga Security Systemの1つの応用例，プロクシについて議論してきた．

従来の「プロクシ」のアプローチはさまざまで，プロクシの持つ本質的な性質や利点を理解しにくくなっており，その有用性を十分に生かすことが困難である．そこで，本論文では，今まで提案されてきた「プロクシ」の本質をとらえた概念を明確に定義し，その特徴や利点について議論してきた．このようなプロクシをセキュリティへ応用したとき，非常に柔軟で高度なセキュリティ制御を提供することができるが，残念ながら従来の認可モデルでは，それを十分に生かすことができなかった．しかし，Saga認可モデルにおいては，

☆ Saga 認可モデルを利用してこの章で述べたようなシステムを実現するためには，サービスパスの完全性（integrity）をはじめとして，従来のセキュリティ機構よりも高度なセキュリティ機構が必要になる．このようなセキュリティ機構については，文献2)，3)を参照されたい．

そのようなプロクシによるセキュリティ制御を単なる応用例の1つとして表現することができる．このことは，Saga認可モデルの強力さ，柔軟性を示すものといえる．

## 参考文献


1) 上園忠弘：ダウンサイジング下における情報セキュリティ，電子情報通信学会，Vol.77, No.5, pp.498–506 (1994).
2) Soshi, M. and Maekawa, M.: A New Authorization Model and its Mechanism Using Service Paths in Open Distributed Environments, *Proc. IFIP TC6 WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems*, pp.251–264 (1997).
3) Soshi, M. and Maekawa, M.: The Saga Security System–A Security Architecture for Open Distributed Systems, *Proc. 5th IEEE Workshop on Future Trends of Distributed Computing Systems*, pp.53–58 (1997).
4) Shapiro, M.: Structure and Encapsulation in Distributed Systems: The Proxy Principle, *International Conference on Distributed Computing Systems*, pp.198–204 (1986).
5) Luotonen, A. and Altis, K.: World-Wide Web Proxies, *Computer Networks and ISDN Systems*, Vol.27, No.2, pp.147–154 (1994).
6) 佐藤 豊：プロトコル中継システムDeleGateの開発，電子技術総合研究所研究速報TR-94-17，電子技術総合研究所 (1994).
7) Jajodia, S. and Kogan, B.: Integrating an Object-Oriented Data Model with Multi-Level Security, *Proc. IEEE Symposium on Security and Privacy*, pp.76–85 (1990).
8) Sollins, K.R.: Cascaded Authentication, *Proc. IEEE Symposium on Security and Privacy*, pp.156–163 (1988).
9) Rumbaugh, J.: Relations as Semantic Constructs in an Object-Oriented Language, *OOPSLA '87 Proceedings*, pp.466–481 (1987).
10) Yahalom, R., Klein, B. and Beth, T.: Trust Relationships in Secure Systems – A Distributed Authentication Perspective, *Proc. IEEE Symposium on Security and Privacy*, Oakland, CA, pp.150 164 (1993).
11) Department of Defense: DoD Trusted Computer System Evaluation Criteria, Technical Report DoD 5200.28-STD, DoD Computer Security Center, Fort Meade, MD (1985).
12) 前川 守：ソフトウェア実行/開発環境，岩波書店 (1992).
13) Sandhu, R.S., Coyne, E.J., Feinstein, H.L. and Youman, C.E.: Role-Based Access Control Models, *IEEE Computer*, Vol.29, No.2, pp.38–47 (1996).
14) Bertino, E. and Samatari, P.: Research Issues in Discretionary Authorizations for Object Bases, *Proc. OOPSLA'93 Workshop on Security for Object-Oriented Systems*, pp.183–199 (1993).
15) Schneier, B.: *Applied Cryptography: Protocols, Algorithms, and Source code in C*, 2nd edition, John Wiley & Sons (1996).


(平成 9 年 9 月 22 日受付)
(平成 10 年 1 月 16 日採録)

**双紙 正和**

昭和 43 年生．電気通信大学大学院情報システム学研究科助手．分散環境におけるセキュリティ・モデル，分散オペレーティングシステムの研究に従事．修士（理学）．

**加藤 丈治**

昭和 48 年生．現在電気通信大学大学院情報システム学研究科情報システム設計学専攻博士前期課程に在学中．開放型分散環境下における異なる組織間での共同作業におけるアクセス制御に関する研究に従事．

**前川 守**（正会員）

昭和 17 年生．東京大学理学部情報科学科助教授等を経て，現在電気通信大学大学院情報システム学研究科教授．主として分散システム，ソフトウェア開発環境等の研究に従事．Ph.D.