社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－CSEC－42（37）
2008 / 7 / 25

分散システムにおける自己拡張型RBACの提案

朝倉 義晴† 本田 篤史† 樋口 直志† 才田 好則†

† NEC システムプラットフォーム研究所
〒211-8666 神奈川県川崎市下沼部1753
E-mail: †asakura@cd.jp.nec.com

あらまし 本稿では，サーバ上に作成するリソースに，リソース作成者の意図を反映させたアクセス制御設定を適用するために，RBAC (Role-based access control) をベースとする自己拡張型 RBAC（Self expandable RBAC:SERBAC）を提案する．SERBAC では，リソース作成者の意図をサーバに伝え，サーバはリソース作成者の意図に基づくアクセス制御設定を定義し，作成するリソースに適用する．SERBAC は，アクセス制御設定を動的に定義することで，管理者不在の環境においても柔軟性の高いアクセス制御を実現できる．
キーワード RBAC，アクセス制御，セキュリティ，分散システム

Self expandable RBAC for distributed systems

Yoshiharu ASAKURA†, Atsushi HONDA†, Naoshi HIGUCHI†, and Yoshinori SAIDA†

† System Platforms Research Laboratories, NEC Corporation
1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan
E-mail: †asakura@cd.jp.nec.com

Abstract In this paper, we propose Self expandable RBAC (SERBAC) based on RBAC in order to apply access control rules, which are reflected in a resource creator's intentions, to resources on a server. In SERBAC, a resoruce creator sends his/her intentions to a server by formal descriptions. A server defines access control rules based on the intentions and applies them to resources. SERBAC realizes flexible access control mechanism in a distributed system without administators.
Key words RBAC, access control, security, distributed system

1. 背 景

分散システムでは，ネットワークに接続された複数の端末が，互いの提供するサービスを利用して動作する．他の端末の提供するサービスを利用するときに，その端末上のリソース（例えばファイルなど）にアクセスすることがある．以後，関係をわかりやすくするために，サービスを提供する端末をサーバと呼び，サービスを利用する端末やユーザなどをクライアントと呼ぶ．本報告では，一般家庭内の個人情報端末や家電機器などから構成される分散システムを対象として考える．これらの端末は住所録などの個人情報や個人のスケジュールなどのプライバシデータを含むことも多いため，サーバの提供するリソースをクライアントに制限なしに利用させることは，情報漏洩やプライバシ保護の観点から望ましくない．リソースの利用を制限する技術として，アクセス制御が挙げられる．アクセス制御では，サブジェクト（アクセス主体者，例えばプロセス）がオブジェクト（アクセス対象，例えばファイル）に対して可能な操作（例えば read,write）を制限する．アクセス制御により，サブジェクトはオブジェクトに対して許可された操作のみ実行できる．

アクセス制御モデルの一つとして，RBAC（Role-based access control）[1] が存在する．RBAC では，ロールを介してユーザ(注1)とパーミッションを関連付ける．ここでパーミッションとは，オブジェクトとそのオブジェクトに対する操作集合の組のことである．例えば，パーミッション (foo, {read,write}) は，ファイル foo に対する read,write 操作を意味する．RBAC では，ロールとパーミッション，及び，ユーザとロールを関連付ける．ユーザとロールを関連付けることで，ロールに関連付けられたパーミッションをユーザに与える．RBAC による適切なアクセス制御を実現するためには，ロールやパーミッションの定義とユーザとロールの関連付けを適切に行う必要がある．し

(注1)：RBAC ではサブジェクトに相当する用語としてユーザを一般的に用いる．本報告書のクライアントに相当する．

かし，これらを適切に行うためには，端末内の構成やユーザ毎のオブジェクトへのアクセス可否を熟知する必要がある．そのため，一般的に，端末の管理者がこれらを定義する．

分散システムでは，クライアントがサーバの提供するサービスを利用することで，サーバ上のリソースにアクセスする．家庭内の分散システムにおけるクライアントは，分散システムに属する端末の構成や利用者により様々である．また，サーバ上にリソースを作成する場合，作成したリソースへのアクセスをリソース作成者（すなわちクライアント）が制御したいという要望がある．これは，サーバ上に作成したリソースへのアクセスを制御することで，リソース作成者の意図に反した情報流通を防ぎたいという要望である．この要望を実現するためには，リソース作成者の意図するアクセス制御設定をリソースに適用する必要がある．本稿におけるアクセス制御設定とは，ロールとパーミッションの定義，および，ユーザとロールの関連付けを定義した規則である．サーバの管理者権限をリソース作成者に与えれば，リソース作成者の望むアクセス制御設定を自身で定義し，リソース作成者の意図するアクセス制御設定をリソースに適用できる．しかし，管理者権限を与えることはセキュリティ上安全ではない．このように，端末のアクセス制御設定は，端末の属する分散システムや利用者の要望により多様であり，端末にあらかじめアクセス制御設定を定義しておくことは困難である．また個々の家庭に，適切なアクセス制御設定の定義を要求するのは現実的ではなく，セキュリティ上安全にアクセス制御設定の定義を可能にすることはできなかった．

本報告では上記の課題を解決するために，分散システムを対象とする自己拡張型 RBAC（Self expandable RBAC:SERBAC）を提案する．SERBAC は，家庭などの管理者不在の環境への適用を想定している．SERBAC では，サーバがクライアントの意図を反映するロールやパーミッションを定義し，サーバの定義したロールとクライアントを関連付ける．すなわち SERBAC では，アクセス制御設定を管理者があらかじめ定義するのではなく，サーバが分散システムの状況（クライアントからのアクセス制御設定の要求）に応じて定義する．このように，サーバが分散システムの状況に応じてアクセス制御設定をセキュリティ上安全に自己定義することで，上記の課題を解決する．

2. 課題設定

本報告で提案する SERBAC の目的，及び，その目的を実現するための解決すべき課題について述べる．

2.1 前提条件

SERBAC は，以下の内容を前提としている．

- 分散システムに属する端末は，コネクション型（例えば，TCP など）で接続されている．すなわち，クライアントがサーバの提供するサービスを利用するとき，それらの端末間にコネクションを確立した後にサービスを利用する．
- 管理者の存在しない分散システム（例えば，家庭内で運用される分散システム）を適用対象とする．また，クライアントの特定に必要な情報（例えば，クライアントとクライアントに一意に割り当てる識別子とを対応付けるデータベースなど）はあらかじめ用意してあるものとする．家庭内を例にすると，家族に識別子を割り当てておいて，ホームサーバ上のデータベースに登録済みであるとみなす．
- リソース作成者が，リソースにアクセス可能なクライアントを決定する権限を持つ．企業内のように，機密文書の作成者とアクセス可能なクライアント（言い換えると公開範囲）を決定する権限を持つ人が異なるような場合は対象外とする．
- パーミッションの記述におけるオブジェクトの指定に，ラベルを用いる．オブジェクトにはラベルを一つ付与する．パーミッションにおいてオブジェクトを直接指定するのではなく，ラベルを用いて指定することで，同一のアクセス制御設定を適用したいオブジェクトには，同一のラベルを付与すればよくなる．例えば，file1 と file2 に同一のアクセス制御設定を適用したい場合，これらのファイルに同一のラベルを付与することで実現できる．オブジェクトを直接指定する場合，オブジェクト毎にアクセス制御設定を定義する必要がある．

2.2 目　　的

SERBAC は，以下の 2 つの目的の実現を目指す．

G1: リソースへのアクセスを制御する．

個人情報やプライバシを保護するために，リソースへのアクセスを制御する必要がある．アクセス制御はサブジェクトの素性に基づき実施し，サブジェクトは許可されたリソースにのみアクセスできる．SERBAC におけるサブジェクトは，クライアントである．この場合のクライアントの粒度としては，ユーザ，端末，端末のベンダなどが考えられる．クライアントがサーバの提供するサービスを利用するとき，クライアントとサーバ間でコネクションが確立されているため，クライアントとコネクションが一対一に対応付けられる．

G2: サーバ上に作成するリソースに対し，リソース作成者にサーバの管理者権限を渡すことなく，リソース作成者の希望するアクセス制御設定を適用する．

リソース作成者が，自身の意図するように，作成したリソースにアクセス可能なサブジェクトを制限したいという要望がある．リソース作成者にサーバの管理者権限を渡すことなく，リソース作成者の希望するアクセス制御設定をリソースに適用できるようにすることで，リソースに含まれる情報の拡散をリソース作成者が制御できる．

2.3 分散システムにおけるアクセス制御の課題

前節で述べた目的を実現するために，以下の課題を解決する必要がある．

P1: クライアントの素性の特定

G1 を実現するために，クライアントの素性を特定する必要がある．

P2: リソースに適用するアクセス制御設定の指定

G2 を実現するために，リソース作成者がリソースに適用するアクセス制御設定を指定できる必要がある．そのため，アクセス制御設定を指定する手段が必要となる．

P3: アクセス制御設定の定義

G2 を実現するために，サーバにおいてリソース作成者の意図するアクセス制御設定が未定義の場合，アクセス制御設定を新

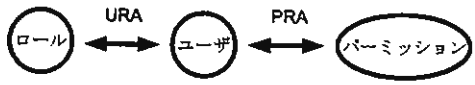

図 1 ユーザ，ロール，パーミッションの関連付け
Fig. 1 Association in RBAC

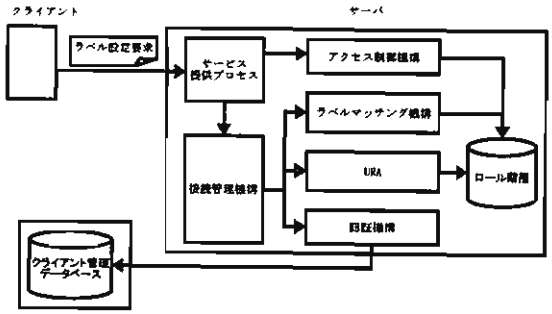

図 2 SERBAC
Fig. 2 SERBAC

規に定義する必要がある．そのため，リソース作成者の意図を形式的に記述する手段とその記述に基づきアクセス制御設定を定義する手段が必要となる．

3. 自己拡張型 RBAC

本報告では，前節で述べた目的を実現するために，RBAC をベースとする SERBAC を提案する．SERBAC では，サーバ上に作成するリソースに対して，クライアントの指定するアクセス制御設定の適用を可能とし，ロールに基づくアクセス制御を行う．

SERBAC は RBAC をベースとしており，前節で述べた 3 つの課題を解決する．

3.1 RBAC

最初に，SERBAC のベースとなる RBAC について述べる．RBAC では，ロールを介してユーザにパーミッションを与える．図 1 に示すように，パーミッションとロールが関連付けられ，ユーザとロールが関連付けられる．ユーザとロールの関連付けは，user-role assignment (URA) と呼ばれ，パーミションとロールの関連付けは，permission-role assignment (PRA) と呼ばれる [2]．RBAC では，ユーザにパーミッションを与えるために，ユーザとロールを関連付けるだけでよい．ロールを介してユーザにパーミッションを与えるため，ユーザとパーミッションとの関連付けを柔軟に行えるという利点がある．RBAC には複数のモデルがあり，その中にロールが階層構造を取る $RBAC_1$ [1] などのモデルがある．階層構造を取るモデルでは，オブジェクト指向のクラス階層と同様に，階層の下位ロールは上位ロールに割り当てられたパーミッションを継承する．

3.2 SERBAC の概要

SERBAC は分散システムにおいて，サービスを提供するサーバに適用される．図 2 に SERBAC の構成図を示す．SERBAC の構成要素は，次の節で述べる．サービスを利用するクライアントは SERBAC の定めるプロトコルに従うことで，クライアントの指定するアクセス制御設定をサーバ上に作成するリソースに適用できる．このプロトコルにおいて，クライアントは，自身の意図を形式的に記述したラベル設定要求をサーバに送信する．サーバは，受信したラベル設定要求に合致するラベルを検索，もしくは，ラベルを定義し，そのラベルをクライアントに返信する．このとき，必要に応じてパーミッションやロールの定義，ロールとクライアントの関連付けも行う．すなわち，アクセス制御設定を定義する．クライアントはリソース作成時にサーバから返信されたラベルを指定することで，作成するリソースに適用するアクセス制御設定を指定する．

次に，SERBAC におけるアクセス制御の概要を述べる．クライアントはサーバとのコネクション毎に認証され，コネクションに対してクライアントの認証結果に応じたロールが割り当てられる．そして，そのコネクションを通したリソースへのアクセスは，そのコネクションに割り当てられたロールに基づいて制御される．

3.3 SERBAC の構成要素

SERBAC は，接続管理機構，認証機構，ラベルマッチング機構，ラベル設定要求，ロール階層，URA，アクセス制御機構の 7 つの要素から構成される．以下，順にこれらの要素について述べる．

（1） 接続管理機構

SERBAC において，クライアントとサーバ間のプロトコルを処理し，コネクションの管理を行う．クライアントからの接続を受け付けたとき，クライアントとのプロトコルを処理し，認証機構を用いたクライアントの認証，ラベルマッチング機構を用いたラベル設定要求に合致するラベルの検索や定義，URA を用いたクライアントとロールの関連付けを行う．

（2） 認証機構

SERBAC ではクライアントの素性を特定するために，クライアントを認証する．認証の手段としては，クライアント認証やパスワード認証が挙げられる．SERBAC では，クライアントを特定さえできれば認証手段を問わない．クライアントの認証後，認証結果と一対一に対応付けられるクライアント識別子をコネクションに割り当てる．ここでクライアント識別子は，分散システム内で一意な識別子とする．認証結果とクライアント識別子との対応付けは，分散システム内に一つ以上存在するクライアント管理データベースにより為される．サーバは，クライアントの認証後，クライアント管理データベースの存在する端末に問い合わせることでクライアント識別子を取得する．そのため，クライアント管理データベースにおいて，あらかじめ認証結果とクライアント識別子を対応付けておく必要がある．

（3） ラベル設定要求

ラベル設定要求は，クライアントの意図を形式的に表現したもので，クライアントとサーバ間のプロトコルにおいて，クライアントからサーバに送信される．サーバはラベル設定要求に合致するラベルを検索，もしくは，定義し，クライアントに返す．この形式的な記法を図 3 に示す．ラベル設定要求は，サブジェクト集合（クライアント識別子の集合）ごとに許可，もし

ラベル設定要求	=	'(' 設定集合 ')'
設定集合	=	(限定子 否定なし操作+) |
		(否定なし操作 | 否定あり操作)+
限定子	=	'only'
否定なし操作	=	'{' サブジェクト集合 操作集合 '}'
否定あり操作	=	'{' 'not' サブジェクト集合 操作集合 '}'
サブジェクト集合	=	'*' | クライアント識別子+
操作集合	=	'{' ('*' | 操作+) '}'
クライアント識別子	=	文字列
操作	=	文字列
文字列	=	文字+
文字	=	英字 | 数字
英字	=	[A..Z] | [a..z]
数字	=	[0..9]

図 3　ラベル設定要求の形式的記法

Fig. 3　Formal form of a label definition request

くは，拒否する操作集合を列挙したもので，列挙した条件を少なくとも満たすようなラベルと合致する．例えば，ラベル設定要求 ⟨{cid1 cid2 {op1 op2}}⟩ は，クライアント *cid1* と *cid2* による操作 *op1* と *op2* が少なくとも許可されるラベルと合致する．この記述に合致するラベルは，他のクライアントによるアクセスや他の操作も許容する．例えば，クライアント *cid3* による操作 *op1* やクライアント *cid1* による操作 *op3* も許可されるラベルに合致し得る．あるサブジェクト集合に対して拒否したい操作集合を指定するときは，not を指定する．また，列挙した条件のみを満たすラベルと合致させたいときは，only を指定する．サブジェクト集合や操作集合にはワイルドカード (*) を指定することも可能である．

この記法に基づくラベル設定要求の例をいくつか示す．

- ⟨only {cid1 {op1}} {cid2 {op2}}⟩

クライアント *cid1* による操作 *op1* とクライアント *cid2* による操作 *op2* のみが許可されるラベルと合致する．この記述に合致するラベルには，クライアント *cid1* による *op1* とクライアント *cid2* による *op2* のみが許可される．他のクライアントによるアクセスや他の操作は拒否される．

- ⟨{not cid1 {op1}} {cid2 {op2}}⟩

クライアント *cid1* による操作 *op1* が拒否され，かつ，クライアント *cid2* による操作 *op2* が少なくとも許可されるラベルと合致する．

- ⟨{not cid1 {op1}} {cid1 {op1 op2}}⟩

クライアント *cid1* による操作 *op1* が拒否され，かつ，クライアント *cid1* による操作 *op1* と *op2* が少なくとも許可されるラベルと合致する．要求が矛盾しているため，このラベル設定要求は無視される．

（4） ロール階層

SERBAC におけるロールは，ルートロールを根とする階層構造（有向無閉路グラフ）を成す．この階層は，$RBAC_1$ における階層と同様に，上位階層のロールと関連付けられているパーミッションを下位階層のロールが継承するという関係を示す．

これらのロールは，ラベルマッチング機構により，クライアント集合と対応付けられる．ロール階層において，下位階層のロールを持たないロールを葉ロールと呼び，下位階層を持つロールを中間ロールと呼ぶ．ルートロールは，上位階層を持たない特別なロールで，ロール階層中に一つ存在する．ロールを新規定義するとき，ロール階層におけるロールの配置位置は，そのロールに対応付けられるクライアント集合により定まる．新規定義ロールは，以下の 2 つの条件を満たす位置に配置される．ここで，ロール r に対応付けられるクライアント集合を $C(r)$ と記述する．

（a） $C(r) \subset C(r')$ を満たす r' の中で，$|C(r')|$ が最小となる r' の下位階層．

（b） $C(r') \subset C(r)$ を満たす r' の中で，重複するロールを除いた(注2)すべての r' の上位階層．

例えば，表 1 に示すようなロールとクライアント集合の対応付けが存在する場合，図 4 に示すロール階層が構築される．

表 1　ロールとクライアント集合の対応付けの例

Table 1　Example of correspondences of a role to a client set

ロール	クライアント集合
ルートロール	cid1,cid2,cid3
role1	cid1
role2	cid2,cid3
role3	cid2
role4	cid3

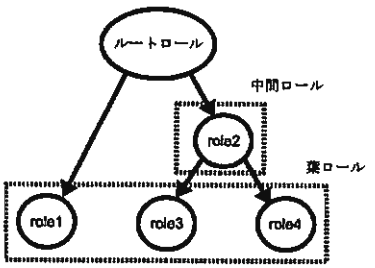

図 4　ロール階層

Fig. 4　Role hierarchy

また，ロールと対応付けるクライアント集合の記述として，除外するクライアントを記述することもできる．例えば，表 1 においてクライアント *cid1* 以外のクライアントを示す集合として，$cid2 \vee cid3$ と $\overline{cid1}$ の両方の記述が可能である．ただし，この両者の記述は新規クライアントをロールと対応付ける場合において同等ではない．例えば，クライアント *cid4* を新たに対応付ける場合を考える．このとき *cid4* は，$cid2 \vee cid3$ と対応付けられるロールとは対応付けられないが，$\overline{cid1}$ と対応付けられるロールとは対応付けられる．

（5） ラベルマッチング機構

ラベルマッチング機構では，ラベル設定要求に合致するラベルを検索，定義する．最初に，ラベル設定要求に合致するラベルが定義済みか検索する．合致するラベルが定義済みの場合，そ

(注2)：ロール r_1, r_2, r が $C(r_1) \subset C(r_2) \subset C(r)$ の関係にあるとき，r_1 は重複するロールである．

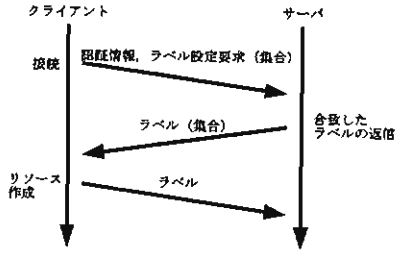

図 5　クライアントとサーバ間のプロトコル
Fig. 5　Protocol between a client and a server

のラベルをクライアントに返信する．合致するラベルが未定義の場合，ラベルとそのラベルに対するパーミッションを定義し，定義したラベルをクライアントに返信する．そして，定義したパーミッションを与えるクライアント集合をラベル設定要求から取得し，そのクライアント集合に対応付けられるロールとパーミッションを関連付ける．このとき，クライアント集合に対応付けられるロールが存在しない場合，ロールも定義する．このようにパーミッションとロールを関連付けることで，クライアントの意図するアクセス制御設定を実現するラベルを定義できる．

（6） URA

SERBAC の URA では，クライアントとロールを関連付けるとき，そのクライアントを含むクライアント集合と対応付けられる葉ロールと関連付ける．なぜなら，ロール階層の構築方法より，クライアント集合と対応付けられるロールは葉ロール，および，葉ロールの上位階層のロールとなるからである．そのため，クライアントに与えるパーミッションという観点からは，葉ロールと関連付けるだけで十分である．

（7） アクセス制御機構

クライアントがサービスを利用する過程で，リソースへのアクセスが発生する．アクセス制御機構は，クライアントとサーバ間のコネクションに割り当てられたロールに基づいてリソースへのアクセスを制御する．

3.4 クライアントとサーバ間のプロトコル

クライアントとサーバ間のプロトコルを図 5 に示す．クライアントは，サーバへの接続時に認証情報(注3)とラベル設定要求（集合）をサーバに送信する．サーバは，ラベル設定要求と合致するラベル（集合）をクライアントに返信する．このようなプロトコルを通して，クライアントは自身の意図をサーバに伝え，意図と合致するラベルを知ることができる．そして，クライアントがサーバ上にリソースを作成するときに，リソースに適用したいラベル（ラベル設定要求と合致したラベル）を指定する．指定したラベルがリソースに適用されることで，クライアントの意図するアクセス制御設定がリソースに適用される．

3.5 SERBAC の処理の流れ

SERBAC は以下の手順に従い，コネクションへのロールの割り当てを行う．

（1） 接続管理機構が，クライアントからのコネクションを受け付ける．

（2） 接続管理機構が，クライアントから認証情報とラベル設定要求（集合）を取得する．

（3） 接続管理機構は，認証機構に認証情報を渡す．

（4） 認証機構は，認証情報を用いてクライアントを認証する．

（5） 認証機構は，認証結果を元にクライアント管理データベースからクライアント識別子を取得する．

（6） 認証機構は，クライアント識別子を接続管理機構に渡す．

（7） 接続管理機構は，ラベルマッチング機構にラベル設定要求（集合）を渡す．

（8） ラベルマッチング機構は，ラベル設定要求（集合）と合致するラベルを検索する．

（9） ラベルマッチング機構は，合致しなかったラベル設定要求（集合）に合致するラベル（集合）とパーミッションを定義し，クライアント集合とロールを関連付ける．関連付けるロールが存在しない場合，ロールも定義する．

（10） ラベルマッチング機構は，ラベル設定要求（集合）に合致するラベル（集合）を接続管理機構に渡す．

（11） 接続管理機構は，URA にクライアント識別子を渡す．

（12） URA はクライアント識別子と関連付けられるロールのロール識別子（ロールを特定するための識別子）を接続管理機構に渡す．

（13） 接続管理機構は，ロール識別子の示すロールをコネクションに割り当てる．

（14） 接続管理機構は，ラベル設定要求（集合）に合致するラベル（集合）をクライアントに通知する．

（15） クライアントは，リソース作成時にリソースに付与するラベルを指定する．

4. ユースケース

SERBAC のユースケースとして，家庭内に存在する HDD レコーダを例に説明する．HDD レコーダは，番組の録画，再生，削除などのサービスをクライアントに提供する．これらのサービスでリソースにアクセスするときに，アクセス制御が適用される．

4.1 環　　境

本事例における環境を述べる．

- HDD レコーダを使う家族は父，母，子供の 3 人で，この 3 人を本ユースケースのクライアントとみなす．
- 父，母，子供の認証結果とクライアント識別子（それぞれ，*fid*, *mid*, *cid* とする）は，家庭内ネットワークに存在するホームサーバ上のクライアント管理データベースにおいて，すでに対応付けされているとする．
- HDD レコーダにおいて，番組はファイル形式で実現される．このファイルを録画ファイルと呼ぶ．
- HDD レコーダは，録画ファイルに対する操作として，再

(注3)：例えば，クライアント認証の場合はクライアント証明書，パスワード認証の場合はパスワード，のように認証の手段により異なる．

生用の play，録画用の record，削除用の remove を定義する．

- HDD レコーダの提供するサービスは，再生サービス，ストリーミング再生サービス，録画サービス，削除サービスである．各サービスを処理するために，再生サービスとストリーミング再生サービスは play を，録画サービスは record を，削除サービスは remove を必要とする．
- HDD レコーダは，録画ファイルに付与するデフォルトのラベルとして，label_any を定義する．録画サービスにおいてクライアントがラベルを指定しなかった場合，録画ファイルにはデフォルトのラベルが付与される．label_any の付与された番組を，ここではパブリック番組と呼ぶ．
- 初期状態において，HDD レコーダの定義するロールは，ルートロールのみである．ルートロールには，パーミッション (label_any, {play, record, remove}) が関連付けられている．また URA において，ルートロールは *cid*，*mid*，*cid* と関連付けられている．

4.2 使用事例

家族による HDD レコーダの使用事例を時系列順に述べる．以下の事例では，携帯電話などのポータブル端末やネットワーク対応テレビをクライアントとして想定している．

事例 1： 父によるパブリック番組の録画

父がパブリック番組を録画する事例を考える．この事例では録画する番組に対してアクセス制御設定の適用を要求しないため，ラベル設定要求の送信は不要である．番組は以下の手順で録画される．

（1） HDD レコーダに接続する．

（2） HDD レコーダ内の認証機構は，クライアントを認証する．そして，認証結果をホームサーバに送信し，認証結果と対応付けられるクライアント識別子 *fid* を取得する．

（3） コネクションに *fid* と関連付けられるロールを割り当てる．今 URA において，*fid* はルートロールと関連付けられているため，ルートロールをコネクションに割り当てる．

（4） クライアントが録画サービスを利用してパブリック番組の録画を要求する．このとき，録画ファイルに付与するラベルを指定しない．そのため，録画ファイルにはデフォルトラベルである label_any が付与される．

（5） HDD レコーダは録画サービス中の処理として，label_any を持つ録画ファイルの作成を試みる．今，コネクションにはルートロールが割り当てられ，ルートロールにはパーミッション (label_any, {record}) が割り当てられているため，番組の録画に成功する．録画ファイルにはラベル label_any を付与する．

事例 2： 母によるパブリック番組のストリーミング再生

母が，お風呂に設置してあるテレビ（クライアント）で，HDD レコーダ内のパブリック番組をストリーミング再生する事例を考える．この事例では番組の再生を行うだけなので，ラベル設定要求の送信は不要である．番組は以下の手順でストリーミング再生される．

（1） HDD レコーダに接続する．

（2） HDD レコーダ内の認証機構は，クライアントを認証する．そして，認証結果をホームサーバに送信し，認証結果と対応付けられるクライアント識別子 *mid* を取得する．

（3） コネクションに *mid* と関連付けられるロールを割り当てる．今 URA において，*mid* はルートロールと関連付けられているため，ルートロールをコネクションに割り当てる．

（4） テレビがストリーミング再生サービスを利用してパブリック番組の再生を要求する．

（5） HDD レコーダはストリーミング再生サービス中の処理として，パブリック番組の録画ファイル（label_any の付与された録画ファイル）のストリーミング再生を試みる．今，コネクションにはルートロールが割り当てられ，ルートロールにはパーミッション (label_any, {play}) が割り当てられているため，番組のストリーミング再生に成功する．

事例 3： 父による子供の操作が不可能な番組の録画

父が，子供による操作のできない番組を録画する事例を考える．この事例では子供による番組への操作を禁止するために，録画ファイルに適用するアクセス制御設定の指定が必要である．番組は以下の手順で録画される．

（1） HDD レコーダに接続する．クライアントはラベル設定要求として，⟨{not cid {*}}⟩ を HDD レコーダに送信する．

（2） HDD レコーダ内の認証機構は，クライアントを認証する．そして，認証結果をホームサーバに送信し，認証結果と対応付けられるクライアント識別子 *fid* を取得する．

（3） HDD レコーダ内のラベルマッチング機構は，ラベル設定要求と合致するラベルを検索する．今，ラベル設定要求と合致するラベルは存在しないため，ラベル設定要求と合致するラベル label_c_exclusion を定義する．そして，ラベルにアクセスするためのパーミッションとして，(label_c_exclusion, {play, record, remove}) を定義する．さらに，$\overline{cid}$ と対応付ける葉ロールとして，非子供ロールを定義する．そして，URA において，$\overline{cid}$ と非子供ロールが関連付けられ，*fid*，*mid* とルートロールの関連付けが削除される．非子供ロールは，ルートロールの下位階層のロールとして定義する．定義したパーミッションは非子供ロールと関連付ける．

（4） コネクションに *fid* と関連付けられるロールを割り当てる．今 URA において，*fid* は非子供ロールと関連付けられているため，非子供ロールをコネクションに割り当てる．

（5） クライアントに，ラベル設定要求に合致するラベルとして，label_c_exclusion を返す．

（6） クライアントが録画サービスを利用して番組の録画を要求する．このとき，録画ファイルに付与するラベルとして，label_c_exclusion を指定する．

（7） HDD レコーダは録画サービス中の処理として，label_c_exclusion を付与する録画ファイルの作成を試みる．今，コネクションには非子供ロールが割り当てられ，非子供ロールにはパーミッション (label_c_exclusion, {record}) が割り当てられているため，番組の録画に成功する．録画した番組にはラベル label_c_exclusion を付与する．

事例 4： 子供による操作が不可能な番組の再生

子供が，label_c_exclusion の付与された録画ファイル（すなわ

ち，子供による操作のできない番組）を再生する事例を考える．この事例では番組の再生を行うだけなので，ラベル設定要求の送信は不要である．番組は以下の手順で再生される．

（1） HDD レコーダに接続する．

（2） HDD レコーダ内の認証機構は，クライアントを認証する．そして，認証結果をホームサーバに送信し，認証結果と対応付けられるクライアント識別子 *cid* を取得する．

（3） コネクションに *cid* と関連付けられるロールを割り当てる．今 URA において，*cid* はルートロールと関連付けられているため，ルートロールをコネクションに割り当てる．

（4） クライアントが再生サービスを利用して番組の再生を要求する．

（5） HDD レコーダは再生サービス中の処理として，label_c_exclusion の付与された録画ファイルの再生を試みる．今，コネクションにはルートロールが割り当てられ，ルートロールにはパーミッション (label_c_exclusion, {play}) が割り当てられていないため，番組の再生に失敗する．

5. 考察と関連研究

5.1 課題への対処

SERBAC は 2.3 で述べた 3 つの課題を，以下に述べる手段で解決している．SERBAC の認証機構は，クライアントを認証することで P1 を解決している．SERBAC のラベル設定要求は，クライアント（リソース作成者）の意図するアクセス制御設定を記述する．そして，SERBAC のラベルマッチング機構は，ラベル設定要求に合致するラベルを検索，定義することで P3 を解決している．SERBAC は，ラベル設定要求に合致するラベルをクライアントに返し，クライアントはリソース作成時にリソースに付与したいラベルを指定することで P2 を解決している．

5.2 有 用 性

分散システムにおいて，サーバ上に作成するリソースに対し，クライアントの意図するアクセス制御設定を適用したいという要望がある．一般的に，アクセス制御設定は端末の管理者によって定義され，リソースに対して適用するアクセス制御設定は，管理者やそのリソースの作成者によって決定される．分散システムでは，分散システムに参加する端末の種類やそれらの端末の使用者により，様々なアクセス制御設定が要望される．そのため，要望に適するアクセス制御設定の定義を管理者に依頼するか，もしくは，分散システムで要望されるアクセス制御設定をあらかじめ定義しておく，のいずれかの対応が必要となる．しかしこれらの対応は，管理者が不在で様々な端末や利用者の介在する分散システム（例えば，家庭内における分散システム）では困難である．SERBAC は，作成するリソースに適用したいアクセス制御設定を，管理者を介さずに動的に定義することでこの問題を解決している．SERBAC は，このような分散システムにおいて有用である．

5.3 情報フロー制御

ここでは SERBAC の情報フロー制御について考察する．ラベルを用いたアクセス制御では，あるラベルに対して，誰によるどのようなアクセスが可能かにより情報フローが定まる．すなわち，サブジェクトに割り当てられるパーミッションに着目する．パーミッションとそのパーミッションの割り当てられるサブジェクトが静的に定まっている環境では，情報フローも静的に定まる（covert channel による情報フローはここでは考えない）．しかし SERBAC では，ラベルやパーミッションが動的に増加し，クライアントに割り当てられるパーミッションもロールを介して動的に増加する．そのため，分散システムの動作中に情報フローも拡張し得る．例えば，クライアント A，クライアント B，クライアント C を考える．また，クライアント A とクライアント B 間にラベル A を介して情報フローが存在すると仮定し，クライアント B とクライアント C 間，クライアント A とクライアント C 間には情報フローが存在しないと仮定する．このとき，クライアント B とクライアント C がアクセス可能なラベル B を作成すると，クライアント B とクライアント C 間に新たな情報フローが発生する．また，クライアント B を介してクライアント A とクライアント C 間にも新たな情報フローが発生する．

SERBAC では，このような情報フローの拡張に対して技術的に対処しておらず，情報フローを完全に制御できない．SERBAC の情報フロー制御は現在のところ，Social Networking Service が参加者を信頼することに依存するように，各クライアントを信頼することに依存している．SERBAC の想定適用対象では，厳密な情報フロー制御を必要としないことも多い．SERBAC の情報フロー制御は，ラベルへのアクセスを許可するクライアントを信頼し，リソースへのアクセスを許可するという信頼性に基づいている．good-enough security [3] に基づく観点と SERBAC の適用対象より，筆者らは，情報フローを完全に制御できなくとも，SERBAC は実用的であると考えている．

5.4 サーバプロセスへの適用

本稿では，SERBAC のサブジェクトとしてクライアントを対象として述べてきた．しかし，サーバ内のプロセスも，SERBAC のサブジェクトとして考えることもできる．この場合，SERBAC はプロセスにロールを割り当て，プロセスに割り当てられたロールに基づきリソースへのアクセスを制御する．例えば，プロセスに割り当てるロールは，以下の手順で決定できる．

（1） プロセスの起動時に，何らかの方法でサーバ内で一意な識別子をプロセスに割り当てる．割り当てる識別子の決定方法として，プロセスのフルパス名に基づく方法，プロセスの実行ファイルに付与されている署名に基づく方法（例えば，DigSig [4]）などが考えられる．なお，プロセスと識別子との関連付けを行うデータベースは各端末が保持する．

（2） URA においてプロセスに割り当てられた識別子と関連付けられるロールを，プロセスに割り当てる．なお，プロセスの識別子とロールとの関連付けをあらかじめ定義しておく必要がある．

サーバ内のプロセスによるリソースへのアクセスも SERBAC で制御することで，リソースへのアクセス制御を一元化できる．

5.5 ラベル設定要求の作成補助 GUI の必要性

ラベル設定要求の記述には，クライアント識別子が必要となる．クライアント端末を使用するユーザに，個々のクライアント識別子を把握させるのは現実的ではない．現実的な解法としては，例えば，具体的なクライアント（例えば，父，母など）とクライアント識別子とを結びつける GUI などを，クライアント端末が提供することである．例えば，リソースへのアクセスを許可するクライアントをリストから選択できるような，ラベル設定要求を作成するための GUI などである．これを実現するためには，クライアント管理データベースにクライアント識別子の意味（例えば，このクライアント識別子は父であるなど）も格納する必要がある．このような情報がクライアント管理データベースに格納されれば，クライアント端末はクライアントとクライアント識別子の組をクライアント管理データベースから取得し，GUI のリストなどに表示できるようになる．

5.6 関連技術，研究

Discretionary Access Control (DAC) は，ファイル管理などで用いられているアクセス制御モデルである．DAC では，リソースの所有者がそのリソースのアクセス制御設定を決定できる．SERBAC ではリソースの所有者という概念はなく，リソースの作成者がリソースのアクセス制御設定を決定でき，この決定はリソース作成時にのみ許可される．

DAC を実現する技術の一つに，Access Control List (ACL) がある．ACL ではリソースに対して，許可する操作集合をサブジェクト毎に割り当てる．そのためサブジェクトの変更（ユーザやグループの増加など）に伴い，全リソースの ACL を更新する必要がある．SERBAC の元となる RBAC では，パーミッションとロールを PRA で関連付け，サブジェクトとロールを URA で関連付ける．そのため，サブジェクトの変更に伴い，URA におけるサブジェクトとロールの関連付けを更新するだけでよく，サブジェクトとパーミッションの関連付けの管理が単純化される．

Digital Rights Management (DRM) はコンテンツの無制限なコピーや利用を防止し，コンテンツ供給者の意図するコンテンツの利用を実現するための技術である．DRM では一般的にコンテンツを暗号化し，正当なユーザの所有する復号鍵で復号することでコンテンツの利用を可能にする．このように DRM ではコンテンツの更新に伴い，コンテンツの再暗号化が必要となる．また，DRM による暗号化は特定の端末と関連付けはされない(注4)．SERBAC では，リソースに付与するラベルに基づきアクセス制御を行う．そのため，リソースの内容の更新に伴い必要な作業はない．しかし，ラベルはサーバ毎に独立であるため，リソースを他のサーバにコピーした場合，コピー先のサーバにおいて適切なラベルを新たに付与する必要がある．

現在，分散システムに適用する RBAC モデルがいくつか提案されている [5]〜[7]．[5] は，イントラネット環境への適用を想定した I-RBAC を提案している．I-RBAC では，端末内に適用されるローカルロールとイントラネット内に適用されるグローバルロールを定義し，これら 2 種類のロールを関係付けてアクセス制御を実現する．[6] は，各端末に定義されるロールとユーザとの割り当てをデリゲーション（delegation）に基づき行う dRBAC を提案している．dRBAC では，ユーザの認証に Public Key Infrastructure を使用している．[7] は，購読モデル（subscription model）に基づいたリソースの利用を分散ロールを用いて制御する DRBAC を提案している．DRBAC では，リソースを利用する組織の定義するローカルロール（URA を制御）とリソースを提供する組織の定義するローカルロール（PRA を制御）を分散ロールを介して結びつける．これらは分散システムを対象とするモデルであるが，各端末におけるロールやパーミッションは管理者があらかじめ定義する必要がある．

(注4)：ただし，復号鍵が端末と関連付けされていない場合．

6. まとめと今後の課題

本報告では，分散システムにおけるアクセス制御モデルとして，RBAC に基づく SERBAC を提案した．SERBAC は，家庭内などの管理者不在の分散システムへの適用を想定している．SERBAC は，クライアントがサーバ上に作成するリソースに対して，クライアントの意図するアクセス制御設定の適用を実現する．クライアントの意図をラベル設定要求として形式的に記述し，サーバはラベル設定要求から必要に応じてラベル，ロール，パーミッションを定義する．そして，クライアントはリソース作成時に作成するリソースに適用するラベルを指定することで，リソースに適用するアクセス制御設定の指定を実現する．リソースに対するアクセスは，クライアントとサーバ間のコネクションに割り当てるロールに基づいて制御される．コネクションに割り当てるロールは，クライアントの認証結果に基づいて決定する．SERBAC は，サーバがクライアントの意図に基づくアクセス制御設定を動的に定義することで，管理者不在の環境においても柔軟性の高いアクセス制御を実現できる．

今後の課題として，SERBAC を適用したプロトタイプを作成し，目的に対する SERBAC の妥当性，実現性，利用性を評価することが挙げられる．

文　献

[1] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman: "Role-based access control models", IEEE Computer, **29**, 2, pp. 38–47 (1996).

[2] R. Sandhu, V. Bhamidipati and Q. Munawer: "The arbac97 model for role-based administration of roles", ACM Trans. Inf. Syst. Secur., **2**, 1, pp. 105–135 (1999).

[3] R. Sandhu: "Good-enough security: Toward a pragmatic business-driven discipline", IEEE Internet Computing, **7**, 1, pp. 66–68 (2003).

[4] The DigSig team: "The digsig project", `http://disec.sourceforge.net/docs/digsig.pdf`.

[5] Z. Tari and S.-W. Chan: "A role-based access control for intranet security", IEEE Internet Computing, **1**, 5, pp. 24–34 (1997).

[6] E. Freudenthal, T. Pesin, L. Port, E. Keenan and V. Karamcheti: "drbac: Distributed role-based access control for dynamic coalition environments", ICDCS'02, IEEE Computer Society, pp. 411–420 (2002).

[7] M. Ma and S. Woodhead: "Constraint-enabled distributed rbac for subscription-based remote network services", CIT2006, IEEE Computer Society, p. 160 (2006).