サービス及びクライアントのカテゴライズによる分散オブジェクトアクセス制御

松本 一輝 橋本 篤 吉田 隆一
九州工業大学 情報工学部

オープンな環境である分散オブジェクト指向計算環境で必要となるセキュリティシステムの中で特に、アクセス制御及びそのために必要となるクライアントの認証を行うシステムについて述べる。生成・消滅を繰り返す多数のクライアントを識別するために、クライアントの性質を示す情報である属性を用いたクライアントの認証を行うモデルと、様々なサービスに対してアクセス権を設定するため、サーバのメソッドをセキュリティレベルによって分類して行うアクセス制御のモデルを提案する。実際にそれらのモデルに基づき、分散オブジェクト指向計算環境へ実装し、処理に要する時間を計測することによる評価を行った。

Distributed object access control by categorization of services and clients

Kazuteru Matsumoto, Atsushi Hashimoto and Takaichi Yoshida
Kyusyu Institute of Technology

Openness, a key characteristics of a distributed object-oriented environment, raises the need of a more secure system. This paper describes access control through client authentication. We propose two models; one is the client authentication model which can identify many short lived clients with attributes that describe client objects properties, the other is the access control model that means many kinds of services are set to access right with categorized server methods according to security levels. Further, a system has been implemented based on those models and a we evaluate its processing time.

1 はじめに

ネットワークを介して複数台のコンピュータが協調しながら計算を行う分散オブジェクト指向計算環境では、様々な利点が得られる反面その安全性が問題となる。そこで本稿では、セキュリティシステムの中でも特にアクセス制御及びそのために必要となるクライアント認証を行なうシステムについて述べる。

この環境においては一般的に、認証を行う対象の単位はオブジェクトとなる。しかし、オブジェクト単位での認証を考える時問題となるのが、オブジェクトの数と、寿命が一定でないそれらの生成・消滅を一つ一つ追跡することが困難であるということである。従って、これらオブジェクトをクライアントとして、個々をサーバが認証することは大変難しい。また、アクセス制御においては、サーバの持つメソッドはその提供するサービスによって様々であり、数も多くなることが問題となる。そのようなメソッドの一つ一つに対して、アクセス権を設定することは非常にコストがかかる。

それらの問題を解決するため、オブジェクトの性質等を表すオブジェクトの属性を用いてクライアントを分類し、それをサーバのクライアント認証に利用する方法 [6] がある。本論文では、属性を用いたクライアントの分類に加え、セキュリティレベルという概念に基づいた、サーバのメソッドの分類によってアクセス制御を実現する手法を提案する。まず、第2章で分散オブジェクト指向計算環境において発生する様々な問題点を挙げ、その解決手法について説明する。次に、その解決手法を用いた我々のアクセス制御モデルを第3章で提案し、第4章で提案したモデルの実装について説明し、その評価を行った。第5章では、関連研究との比較を行い、最後に第6章で本稿のまとめを行う。

2 問題点とその解決

2.1 クライアントの分類による認証

アクセス制御を行う際、まず、アクセスを行ってきた相手を特定する認証を行う必要がある。しかし、認証を行う単位がオブジェクトとなる分散オブジェクト指向計算環境では、オブジェクトは多数存在し生成と消滅を繰り返している。従って、サーバはクライアントを個々に識別することは非常に困難である。

そこで、オブジェクトの属性を利用してクライアントをカテゴライズし、そのカテゴリでクライアントを認証する方法をとる。ここで言うオブジェクトの属性とは、インスタンス変数の値などのオブジェクトの内部データではなく、オブジェクトの役割や性質、所属等を表す情報である。これは、オブジェクトを分類できればよく、オブジェクト個々を特定するものでなくとも良い。オブジェクトの属性の例としては、オブジェクトを生成した時刻やその所有者、クラスの設計者、そのオブジェクトのサービスの内容等が考えられる。また、オブジェクトが何らかのプロジェクトに所属している場合やグループを形成している場合、それらを属性とすることができる。

クライアントはこの属性をあらかじめ付与される。そして、サーバはそのアクセスをおこなったオブジェクトの属性が正当なものによって与えられた属性であるかどうか(正当性)と、改竄されていないこと(完全性)を検証することで、クライアントの認証を実現する。

オブジェクトの属性を利用することによって、サーバはクライアント個々ではなく、図1の様に分類されたクライアントのグループという単位をアクセス制御の対象とすることが可能となり、いくつかの属性を組み合わせて指定することで、柔軟なクライアントの分類が行える。これによって、環境に多数存在するオブジェクト個々を識別する必要がなくなり、オブジェクトの生成や消滅をサーバが追跡する必要がなくなるという利点を得る。

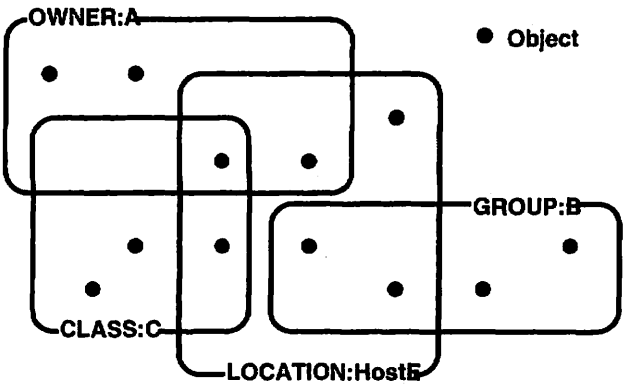

図 1: 属性を利用した認証

2.2 メソッドの分類によるアクセス制御

オブジェクト指向計算環境においては、サーバの状態を操作することのできるメソッドへのアクセス権こそが制御すべきアクセス権となる。そこで、サーバに対するアクセス制御の単位をメソッド単位とする。しかし、あるまとまったサービスを提供するサーバの持つメソッドは、通常、その数が多くなり、サーバの生成者がそれらのメソッド個々に対してアクセス権を設定していくことは、非常にコストがかかる。従って、サーバの持つメソッドを分類して扱えるようにしたい。しかし、そのメソッドの種類も提供するサービスによって様々であり、あらかじめ予想することが困難である。

そこで本研究では、セキュリティレベルによってメソッドを分類し、この分類されたカテゴリに対してアクセス権の設定を行う。Unix 等ではファイルのアクセス権を設定することができ、そこでは read よりも write がファイルに与える影響が大きいため、アクセス権を高く設定する。それと同様に、クライアントに操作されるメソッドがサーバの状態にどのような影響を与えるのかを考え、その影響の大きさをレベル分けしたセキュリティレベルをメソッドに設定する。メソッドは、サーバの状態を変更も公開もせず、単に与えられた引数から決められた計算を行い、その結果を返すメソッドから、オブジェクトの状態を書き換えてしまうものまで様々であり、それはいくつかに分類することができる。例えば、スタックを管理するオブジェクトにおいては、管理しているスタックの大きさの上限の値を返すようなメソッドよりも、スタックの保持している値をクリアするようなメソッドの方がセキュリティレベルを高く設定する必要がある。

セキュリティレベルに従ってメソッドを分類することで、アクセス権の設定をまとまった単位で行なうことが可能になる。このセキュリティレベルによっ

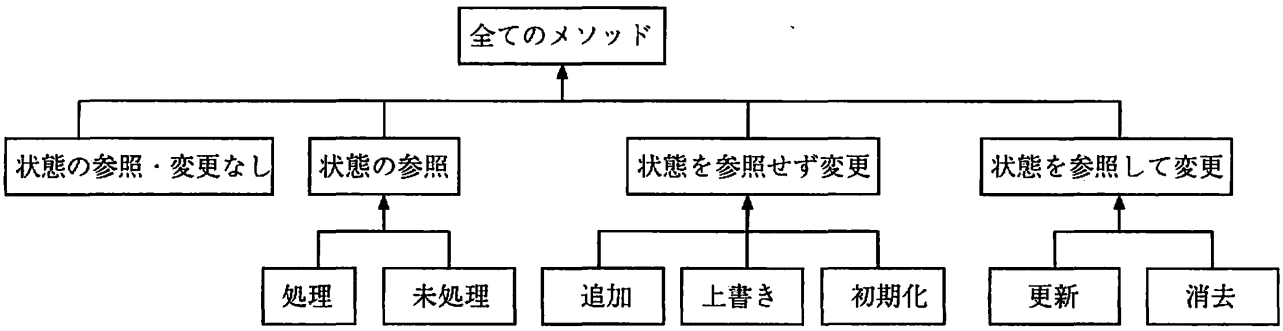

図 2: カテゴリ階層

てまとめられたメソッドの集合を、ここではカテゴリと呼び、入れ子構造の階層となる。上位のカテゴリは、それよりも下位にある全てのカテゴリに含まれる全てのメソッドを含む。カテゴリの階層化によって、メソッドの分類を柔軟に行うことができ、サーバでのアクセス権の設定を簡潔に行なうことができる。

3 モデル

3.1 属性の付与と認証

属性を用いた認証では、属性の付与と属性の認証とを順に行なう必要がある。また、属性の付与や認証の際には公開鍵暗号系を用いた署名が行われる。そのための公開鍵を管理する機構も必要となる。

属性付与 ある属性について権威を持つ人間に代わって属性付与者がクライアントに対し、属性を付与する。属性付与者は、クライアントがサーバへアクセスする前に公開鍵暗号を用いた電子署名を施した属性をクライアントに付与する。事前に電子署名に使われる秘密鍵に対応する公開鍵を公開鍵管理者へ登録しておく。この登録された鍵を用いて属性の認証を行う。

属性認証 クライアントに付与された属性が不正に改竄されてないことを検証し、認証する。属性認証者はオブジェクトが持つ属性を全て属性付与者の公開鍵で認証し、その正当性、完全性を確認できれば属性証明書をクライアントに対して発行する。この証明書を持つことでクライアントが持つ属性がサーバに対し有効になる。この属性認証者によって、サーバは属性認証者のみを信用すれば良いことになる。

属性の認証を個々のサーバで行うことはコストが高い。そこで、アクセス対象となるサーバとは別に、オブジェクトの属性の認証を行う属性認証者を用意する。クライアントがサーバへアクセスする際には属性そのものではなく属性証明書をメッセージと一緒に渡す。サーバは渡された属性証明書の署名を検証し、検証に成功したら、証明書の中の属性をアクセス制御の際に用いる。証明書は、クライアントの識別子とクライアントが持つ全ての属性名と属性値、属性証明書を発行した属性認証者の名前と属性証明者が自らの署名によって成る。

3.2 アクセス制御

属性によって認証されたクライアントに対して、サーバのどのメソッドへのアクセスを許可するかを決定する。そして、メソッドをセキュリティレベルによってカテゴライズし、そのカテゴライズされたメソッドのカテゴリごとにアクセス権を設定する。そのアクセス権の設定は、クライアントの持つ属性の組み合わせによって行われる。

メソッドの分類 セキュリティレベルによるサーバのメソッドの分類は、図 2 の様な階層化されたカテゴリに従って行われる。メソッドの性質を、オブジェクトの状態の参照の有無と、状態の変更の有無によって大きく 4 つにカテゴライズした。そして、それぞれのカテゴリに対してさらに細かいサブカテゴリを設定した。例えば、サーバの状態に対して一切の参照と変更を伴わない、クライアントに対して計算の結果だけを返すようなメソッドは、一番左側のカテゴリに含まれる。また、個別の値を参照できるメソッドよりも、統計処理を行った値を返すようなメソッド

の方がセキュリティレベルは低い。上書きカテゴリと更新カテゴリに含まれるメソッドの違いは、サーバの状態を参照するかどうかの違いである。上書きはサーバの状態を参照せず、ただ同じフィールドに対して値を書き換え、更新は状態を参照した上で値を書き換える。そして、階層の根であるカテゴリへのアクセスを許可すれば、全てのソッドへのアクセスをクライアントへ許可できる。

また、サーバのサブクラスでは、継承したメソッドをオーバーライドするか明示的にカテゴリを指定しない限り、メソッドの属するカテゴリは引き継がれる。

アクセス権の設定 アクセス制御を行なうための設定として必要なものは、サーバの持つメソッドを分類したカテゴリの設定と、そのカテゴリにアクセスを許可するクライアントの設定である。これらの設定を記述することで、アクセス制御を行う。アクセス制御の設定をサーバを構成するクラスのプログラムコードから分離する。これは、プログラムコードとアクセス制御の設定の両方の可読性を向上させ、異なるサーバへの再利用が望める。

ケイパビリティ クライアントがサーバへアクセスを行う度に、クライアントのアクセス権を検査することは時間的に効率が悪い。この問題を解決するため、本研究ではAmoebaのような分散オペレーティングシステムで利用されているケイパビリティ[1]をサーバが認証を終えた後に、クライアントへ発行する。以降は、発行されたチケットをサーバへのアクセスのために提示し、冗長な処理を省いて効率を上げることができる。これは共通鍵暗号系で暗号化される。

3.3 処理の流れ

1. クライアントは属性を属性認証者に付与してもらう。クライアントは属性を複数持つことが可能で、それぞれ対応する属性認証者が存在する。
2. 付与された属性の正当性と完全性を検証し、それを他者(サーバ)に対して証明するのが属性認証者である。属性認証者は、複数の属性の認証に対して責任を負い、クライアントへ属性証明書を発行する。
3. 属性証明書を提示してサーバへアクセスを行い、サーバのメソッドのアクセス権を得る。

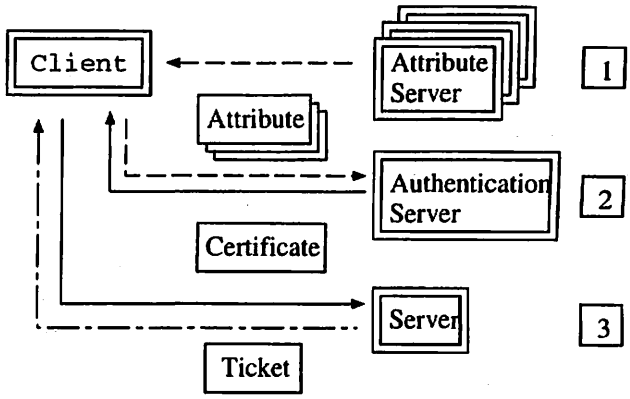

図 3: 処理の流れ

図3はアクセス制御の処理の流れを示している。サーバでは、サービスを提供するためのメソッドをあらかじめセキュリティレベルによって分類しておく。また、その分類されたカテゴリに対して、どの属性の組でアクセスを許可するかを設定する。そして、クライアントがアクセスを行った際に、そのクライアントが提示した属性証明書に記述されている属性によって、カテゴリのメソッドへのアクセスの許可／不許可を決定する。

4 実装と評価

本研究で提案したモデルを実際に分散オブジェクト指向環境へ実装し、その処理時間による評価を行う。実装の対象として、現在我々が開発を行っている適応型分散オブジェクト指向計算環境 Juice[4] を用いる。

4.1 実装

提案したモデルに従い、属性付与者としての役割を持つ属性付与サーバと、属性認証者としての役割を持つ属性認証サーバを Juice へ実装した。属性認証サーバには、公開鍵暗号を用いて属性や属性証明書に署名をするために用いる公開鍵管理者の代理としての機能も有する。これは、鍵の受渡しのコストを低減させるためである。

```
class Test;

import newPackage.abc;
import java.util.*;

Independent{
 print();
 load(String f_name);
}
Refer{
 get(String[], Object);
 get(double, Vector, abc);
}
```

図 4: カテゴリに含まれるメソッドの記述

```
Independent{
 CLASS(Foo);
 OWNER(kazu,taka);
}
Refer{
 GROUP(*);
 LOCATION(jp,uk,usa);
 SIZE(<100MB);
}
```

図 5: 属性に許されたアクセス可能なカテゴリの記述

ここで使用される電子署名技術には、DigitalSignature Algorithm (DSA) [2] を用いた。そして、クライアントに対してサービスを提供するサーバに、そのサーバのメソッドのアクセス制御機構を実装する。それに伴いアクセス制御を行うための設定の記述方法も決定した。まず図 4 に示すように、サーバのメソッドがどのカテゴリに含まれるかを記述したファイルを用意する。この設定ファイルはクラス毎に用意され、クライアントがアクセス可能なメソッドの全てがカテゴライズされる。次に図 5 のような、どの属性を持つクライアントがどのカテゴリにアクセス可能かを示したファイルを用意する。ここでは、ワイルドカードの使用や、サイズの範囲指定などを行なうことが可能である。以上のファイルよりサーバはテーブルを作成し、そのアクセスしてきたクライアントの持つ属性に対して設定されたアクセス権を検査する。

設定記述をコードと分離することで、設定ファイルの再利用性と可読性の向上が望める。また、設定ファイルを変更することで、実行時にアクセス制御の設定を更新することが可能である。

Case	Time(msec)
属性付与 (属性 1 個のみ)	200
属性認証 (属性 1 個のみ)	994
サーバへのアクセス (初回)	391
サーバへのアクセス (2 回目以降)	5.2
署名の検証	388

表 1

4.2 評価

属性の認証とサーバにおけるアクセス制御のそれぞれの処理に要する時間を計測した。最適化処理なしで 1000 回連続で行った平均時間を表 1 に示す。

これによると、クライアントが属性 1 つを属性付与サーバに付与してもらうためには 0.2 秒、属性認証サーバに属性証明書を発行してもらうために 1 秒程度が必要となる。さらに、サーバへの初回のアクセスには 0.4 秒近くかかっている。ここで属性の付与とその認証、さらに属性証明書に利用している公開鍵暗号による署名の作成と検証に要する時間の計測した。これを見ると、実行時間のほとんどが署名検証、署名の作成に費やされていることがわかる。つまり本アクセス制御機構において公開鍵暗号による署名の作成、検証が処理速度低下ののボトルネックになっていると言える。

しかし、サーバへのアクセスを行うと、クライアントはサーバからケイパビリティを受け取る事ができる。これを以降のアクセス時にサーバへ提示することで、アクセス制御に要する時間を 5ms 程度に押さえることができた。これは、ケイパビリティを電子署名を使わず共通鍵暗号系を使って暗号化したことによる高速化である。属性の付与やその認証によって発行される属性証明書は、クライアントの属性が変更されない限り、最初の一回だけの作成で済む。通常オブジェクト指向計算環境では、クライアントとサーバ間のやり取りは複数回に及ぶ。サーバへのアクセスについても、クライアントがサーバへアクセスする際の初回にだけコストがかかるだけであり、この処理速度低下の問題は公開鍵暗号全般に言えることでもある。そして何より、分散環境においてアクセス制御機構の導入は必要不可欠であることから、このような処理時間の増加は許容できると考える。

5 関連研究

本研究と同様の分散環境向けの認証機構にKerberos[5] がある。認証を個々のサーバで行わずに信用できる第三者によって行っている点では本認証機構と同じである。またKerberosでは、クライアントとサーバ間の相互認証をサポートしている。だが、Kerberosにおけるクライアント認証はオブジェクトの所有者という面でしか認証を行っていない。本研究ではクライアントに属性という情報を持たせ、それを認証する方法をとった。これによって、アクセス制御の際にオブジェクトの属性の組み合わせをアクセス制御の対象にできるため、より柔軟なアクセス制御が可能となり、属性を用いることによるクライアントのカテゴライズによって、サーバ管理のコストを低減している。しかし本研究では、クライアントとサーバ間の相互認証について考慮していない。これは、今後の課題でもある。

また、本研究と同じような階層化されたアクセス制御機構を提供するものとしてJavaのセキュリティ機構 [3] がある。しかし、それはファイルやネットワークへの接続と言った資源(ターゲット)と、それへのreadやwriteなどのアクションによって行われている。また、オブジェクトのメソッドやクラス、変数の単位でアクセス制御を行う方法として、private・protected・public・指定なしの4つのみでアクセス権を設定する。本研究の提案するアクセス制御機構は、分散オブジェクト指向計算環境への適用を考慮し、クライアント及びサーバのメソッドをそれぞれ属性とセキュリティレベルによって分類し階層化することで行われる。特に、メソッドをアクセス制御の単位としたことで、Javaのセキュリティ機構よりも設定を簡易に行える。しかし、本モデルではカテゴリの階層は固定である。これは開発者による拡張を許さず、セキュリティホールの発生を抑える利点もあるが、Javaのセキュリティ機構のように開発者がアクセス制御の階層に新しいルールを追加できるような拡張性を欠いていると言える。

6 おわりに

本稿では分散オブジェクト指向計算環境上のセキュリティにおけるアクセス制御機構のモデルについて提案した。

クライアントに属性を持たせて認証することにより、サーバは環境に多数存在するクライアントを特定することなく、アクセス制御の対象とすることを可能とした。アクセス制御を行う単位としては、サーバの持つメソッドをセキュリティレベルによってカテゴライズした、階層化されたカテゴリを用意した。これにより、柔軟なアクセス制御の設定が可能となる。アクセス制御の設定でも、サーバを構成するクラスのプログラムコードから分離して記述することで、設定の可読性や再利用性が向上し動的変更が可能であるといった利点が得られた。また、アクセス制御の処理における時間的コストを考慮し、クライアントに対してケイパビリティを発行することとした。これにより、処理における時間的コストを大幅に減少させることができた。

しかし、実装によって以下の問題もまた判明した。まず、どのように様々なクライアントの属性を付与するかという問題がある。また、属性認証者が複数存在するときの信用の委譲の問題や、サーバ、クライアント間の相互認証ができないといった問題も存在する。そして、オブジェクトの移送などにより動的に属性が変化する場合、属性に履歴機能も必要になる。そしてサーバのアクセス制御機構においても、アクセス制御機構の設定を変更するメタレベルでのアクセス制御機構を考慮していなかったため、セキュリティが完全ではないという問題がある。このような問題は、今後解決していかなくてはならない。

参考文献

[1] Andrew S.Tanenbaum: Distributed Operating System: PRENTICE HALL,1995

[2] Digital Signature Algorithm: http://www.itl.nist.gov/fipspubs/fip186.htm

[3] Java セキュリティアーキテクチャ: http://java.sun.com/j2se/1.4/ja/docs/ja/guide/security/index.html

[4] 小田謙太郎: 適応型分散オブジェクト指向環境のためのオブジェクトモデルと実装: 第5回プログラミングおよび応用のシステムに関するワークショップSPA'02, 2002

[5] Kerberos：http://www.rccm.co.jp/ juk/krb/ beginkrb.txt

[6] 藤島朋和: オブジェクトの属性認証によるセキュリティ: マルチメディア, 分散, 協調とモーバイル (DICOMO '99) シンポジウム, 平成11年6月