大学における統一認証基盤としてのCASとその拡張

内藤久資[†] 梶田将司[††] 小尻智子[††]
平野靖[††] 間瀬健二[††]

本論文では，大学における複数の情報システムで共通に利用できるセキュアな認証基盤の実現の1つの方法として，Yale大学で開発されたCentral Authentication Service (CAS) を拡張したCentral Authentication and Authorization Service (CAS^2) について述べる．CASによる認証システムは，個々の情報システムが認証情報を知ることなく認証結果だけを得ることが可能な，標準的なWeb技術だけを用いて実装することができるシングルサインオンサービスである．我々はCASのセキュリティ上の問題点などを指摘し，単なるAuthenticationだけでなくAuthorizationにまでサービスを拡張することによってそれらの問題点を克服した．さらに，CAS^{2}を用いたシステムとして，名古屋大学ポータルおよび学務情報システムの運用結果にも触れ，CAS^{2}のさらなる拡張についても議論する．

CAS and Its Extension as an Unified Authentication and Authorization Information Infrastructure for Institutions

Hisashi Naito,[†] Shoji Kajita,[††] Tomoko Kojiri,[††]
Yasushi Hirano[††] and Kenji Mase[††]

In this paper, we discuss an institutional unified authentication and authorization infrastructure using Central Authentication Service (CAS) developed by Yale University and its extension to Central Authentication and Authorization Service (CAS^2). Since CAS server only treats authentication information, hence CAS provides a Single Sign On service in a more secure manner. Moreover CAS is implemented on typical standard Web technologies so that the required implementation cost is quite low. We also discuss security problems of CAS, and we improve CAS as CAS^2. Furthermore, we explain the deployment of CASified Nagoya University Portal and a Web-based course registration application at 2005 first semester and summarize our experiences with the discussion of further extensions in CAS^2.

1. はじめに

近年の情報システムでは，不特定多数のユーザに対して一様な情報を提供するだけではなく，ユーザを特定して情報を提供したり，個人情報をシステムに蓄積することを求めたりする状況が頻繁に発生している．多くの場合，このような情報システムにおいて，個人を特定するための「認証」は各システム上で独自にインプリメントされることが多い．

高等教育機関における情報システムは，機関内の多様な管理階層を反映して，情報システム自身も多様な管理形態がとられている．たとえば，機関全体で利用するシステムだけでなく，学部・学科などで利用するシステムなどが混在する．また，それらの管理も，学科・学部・各部門など，多様な管理レベル・形態のシステムが混在している．管理形態の多様さは認証システムの多様さに直結し，ユーザにとってはシステムによってユーザIDを使い分ける必要性が生じる場合がある．

このようなシステム管理体系の多様さは，システム運用コストの増大をもたらすだけでなく，可用性やセキュリティの低下をもたらす原因となることが考えられる．そのため，認証データベースの統一化が図られ，全学レベルでの認証基盤の上に立った情報システムの構築が進められることが多いが，一方では，情報サービスのきめ細かさや学部・学科ごとの特殊性をともなう情報システムの存在があり，すべての情報システムの認証系を統一することは，高等教育機関内の多様性を損なう可能性があるため困難をともなう．

統一認証データーベースを利用した場合には，ユー

† 名古屋大学大学院多元数理科学研究科
Graduate School of Mathematics, Nagoya University
†† 名古屋大学情報連携基盤センター
Information Technology Center, Nagoya University

ザ ID やパスワードなどの認証情報を取り扱うなど，多様な管理形態を持つ情報システムが認証データベースの管理下にあるデータに深く関与するため，情報システムのセキュリティ上の問題が認証データベースそのもののセキュリティ上の問題に直結することが考えられる．

しかしその一方では，インターネットなどの情報技術の普及にともなう教育・研究活動での情報技術の活用は不可欠となってきている．このため，各管理階層において多様な情報システムが構築され，教職員だけでなく学生に対しても多くのサービスを提供する必要に迫られている．このように，現在の高等教育機関の情報システムに代表される多様な組織構造を有する大規模組織においては，単なる認証システムの統一化だけでなく，組織構造の多様性に配慮したセキュアな認証システムもしくは認証インタフェースの実現が必要不可欠である．そのような認証インタフェースは，利用者が多様であってもユーザフレンドリであることが求められるとともに，多様な利用者が多様な情報システムを利用するため，シングルサインオン機能の実現も不可欠な要素である．

我々は，高等教育機関における多様な情報システムの認証基盤の統一化を実現するための1つの方法として，Yale 大学による Central Authentication Service (CAS) を拡張し，強力な権限管理機構を持つ Central Authentication and Authorization Service (CAS^2) を開発した．本論文では，CAS^2 を利用した全学認証基盤について述べる．また，CAS^2 を用いて運用を行った実例として，名古屋大学ポータルと学務情報システムを紹介し，平成16年度後期成績入力・平成17年度前期履修登録の実運用結果についてまとめる．

2. Central Authentication Service

CAS は Yale University ITS Technology & Planning によって開発された，主に Web ベースのアプリケーション（以下では，単に「アプリケーション」と書く）に対してシングルサインオン環境を実現するための認証機構である[1),2)]．現在は，Java Architecture Special Interest Group (JA-SIG) のオフィシャルプロジェクトとしてオープンソースで継続的に開発が進められている．JA-SIG uPortal Summer Conference (2005 年 6 月) において，CAS Version 3[3),4)] が発表され，DI (Dependency Injection) コンテナとして最近注目を浴びている Spring Framework による実装に大幅に更新されたが，基本的な動作に変更はない．ここではこれまで利用されてきた CAS Version 2 に基づいて CAS の概要および認証の流れを述べ，CAS Version 2 の問題点について整理する．

2.1 概　　要

CAS は Java Servlet API 2.3 をベースにした Web アプリケーションであり，その特徴は，以下のようにまとめることができる．

- 認証過程においては，Java Script によるリダイレクション・クッキー・URL パラメータなどの標準的かつ基本的な Web 技術だけが利用される．
- CAS による認証を利用するアプリケーションには，CAS クライアントと呼ばれる，CAS サーバとの通信を行うライブラリを組み込む必要がある．CAS クライアントは Java・PHP・Perl・PL/SQL・Python など，アプリケーションで標準的に利用されるプログラム言語で利用できるほか，静的なアクセスコントロールを行う Apache 用モジュールなどがある．
- 認証データ（ユーザ ID・パスワード）はユーザ（Web ブラウザ）から CAS サーバに対してのみ送信され，CAS を利用するアプリケーションには認証データは送信されない．
- CAS サーバ自身は認証データベースを持たず，外部データベースを利用する．外部データベースとしては，LDAP・SQL データベースなど種々の形式に対応可能である．
- 北米を中心に豊富な採用実績（2005 年 6 月現在，30 を超える大学で採用）を持つ．

2.2 認証メカニズム

CAS による認証過程において重要な役割を果たすのは次にあげる 2 つの「チケット」である．

- “Ticket Granting Cookie” (TGC)：TGC とは CAS サーバからブラウザに与えられるクッキーであり，ブラウザが認証済みどうかを判断する．
- “Service Ticket” (ST)：ブラウザが CAS 認証を利用するアプリケーションへアクセスする際の “One-Time Ticket” の役割を果たす URL パラメータ．ブラウザが CAS サーバにアクセスして得る．

これら TGC・ST は CAS サーバ内部のデータベースにも記憶される．ユーザ情報は TGC に付随し，ST をキーにして TGC およびユーザ情報を検索することができる．また，TGC・ST ともに「有効期限」を持ち，TGC のタイムアウトは関連するアプリケーション全体に対する「セッションタイムアウト」として機能する．一方，ST の有効期限は Man-in-Middle attack を防ぐため短い値に設定する．

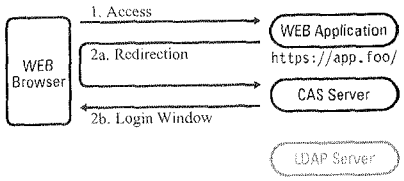

図1 CASの動作(1)
Fig. 1 Action of CAS (1).

CASサーバは以下の2つの基本的なサーブレットから構成される.

- "Login":ブラウザからのアクセスを受理するサーブレットであり,必要であればユーザに「ログイン画面」を提示して認証データを得る.また,TGCおよびSTをブラウザに対して供給する.
- "Validation":アプリケーションからのアクセスを受理するサーブレットであり,STの正当性を検証し,アプリケーションに対してユーザ情報を供給する.

2.2.1 CAS認証(1)

まず,CAS認証を必要とするアプリケーションにユーザが初めてアクセスする際の動作過程を解説する.

(1) アプリケーション(たとえばhttps://app.foo/)にアクセスする(図1-1).初回のアクセスではSTを持たないため,アプリケーション内のCASクライアントライブラリはCASサーバ(Loginサーブレット:https://cas.foo/login)へのリダイレクションを発生する(図1-2a).リダイレクション先のURLにはserviceパラメータと呼ばれるURLパラメータを挿入し再転送先のURLをCASサーバに伝える(https://cas.foo/login?service=https://app.foo/).リダイレクションによるアクセスを受理したLoginサーブレットはブラウザが有効なTGCを持つかどうかを判断し,有効なTGCを持たない場合には,ブラウザに対して「ログイン画面」を提示する(図1-2b).

(2) ユーザからの認証データを受理したLoginサーブレットは外部認証サーバを利用して認証を行う(図2-3).認証に成功した場合には,ブラウザに対してTGCを発行し,serviceパラメータで指定されたURLに対するリダイレクションを発生する(図2-4).リダイレクション先のURLにはticketパラメータの形でSTを挿入する(https://app.foo/?ticket=ST-NNN-XXXXXX).

(3) STをパラメータに含むアクセスを受理したアプリケーションはSTをValidationサーブレットに送付する(図3-5a).Validationサーブレットは STの正当性を検証し,正当なSTであれば,該当のユーザ情報をアプリケーションに送付する(図3-5b).アプリケーションはユーザ情報に従ってブラウザにHTMLデータを送付する(図3-6).

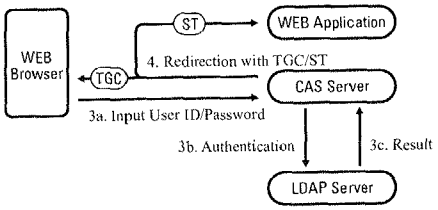

図2 CASの動作(2)
Fig. 2 Action of CAS (2).

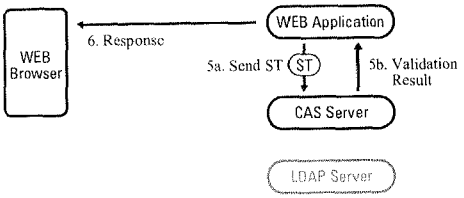

図3 CASの動作(3)
Fig. 3 Action of CAS (3).

2.2.2 CAS認証(2)

有効なTGCを持つブラウザ,すなわち,認証を済ませたユーザがログイン時と同じアプリケーション,または異なったアプリケーションにアクセスする際にも同様の過程を経てレスポンスを得る.

有効なTGCを持つブラウザのアプリケーションへのアクセスの過程において,ログイン時と異なる動作をするのは以下の部分である.

- 有効なTGCを持っているため「ログイン手続き」が省略され,Loginサーブレットからはアプリケーションへの STを含むリダイレクションが発生する.すなわち,2.2.1項(1)のログイン手続きが省略される.

この有効なTGCを持ち他のアプリケーションへ初めてアクセスする場合においてもまったく同様の過程を経る.すなわち,同一のCASサーバを利用するアプリケーションに対してもシングルサインオン環境を提供している.

2.3 アプリケーションのCAS化とCASクライアント

通常のアプリケーションはForm入力などを受理した直後に認証ルーチンを呼び出し,その結果に従って

HTML データを生成することが多い．このような記述を行っているアプリケーション（プログラム）に対してはその CAS 化はきわめて容易である．実際，認証ルーチンを，呼び出し部分を CAS クライアントの呼び出しに変更するだけで CAS 化が実現できる．このような動作を行う CAS クライアントの動作アルゴリズムは以下のとおりである．

アルゴリズム 1（CAS クライアント）

```
if （URL パラメータ内に ticket を含まない） then
  CAS サーバ（Login）へリダイレクト ;
else begin
  ST を CAS サーバ（Validation）へ送付 ;
  Validation 結果を受理 ;
  if （Validation Failure） then
    rc.result := FAIL ;
  else begin
    rc.result := SUCCESS ;
    rc.netid := UserInformation ;
  end
  return rc ;
end
```

2.4 CAS の問題点

このように，CAS を用いることにより容易にセキュアなシングルサインオン環境を構築することができるが，CAS には以下のような問題点が存在する．

(1) GET 入力メソッドにのみ対応でき，POST 入力メソッドに対応していない．

(2) 国際化に対応していない．すなわち，Form の入力に日本語を含むアプリケーションに対応できない．

(3) Validation サーブレットへのアクセス制限に対応していない．

(4) クロスサイトスクリプティング脆弱性が存在し，標準的な手法での TGC の流出が発生しうる．

(1)，(2) は CAS サーバ内部のパラメータ処理の問題であり，実運用上は対応が必要不可欠な機能である（対応方法については 3.4 節を参照）．一方，(3)，(4) はアプリケーションのセキュアな運用のためには，クリアしなければならない重大な問題であるが，我々は CAS に権限管理機構を組み込むことでこの問題をクリアした．特に Validation サーブレットへ無制限なアクセスが許されることは，ランダムな ST を発生し，Validation サーブレットで検証することにより，ユーザ情報が詐取される可能性を含むセキュリティ上の問題となりうる．

3. Central Authentication and Authorization Service

冒頭に述べたように，高等教育機関における統一認証基盤は種々の管理階層によるアプリケーションのセキュリティと高可用性をサポートするものでなければならない．統一認証基盤として CAS を利用するだけでは 2.4 節 (3)，(4) の問題点があるだけでなく，アクセス権を持つユーザの設定・クライアントの制限・アクセス時間の制限などの権限管理機構（Authorization）や二重ログインの防止・セッションタイムアウト・セッション維持機構などを各アプリケーションごとに組み込む必要がある．特に，各アプリケーションに権限管理機構を組み込むためにはユーザ ID などのユーザ識別情報以上のユーザ情報を CAS から受け取る必要があり，「ユーザ認証情報は CAS だけが扱う」という基本ポリシに反する場合がありうる．

我々はこのような権限管理機構を CAS に組み込み，アプリケーションのセキュリティと高可用性をサポートすることに成功した．以下では権限管理機構を組み込んだ CAS，すなわち CAS^2 の権限管理機構とその管理方法について述べる．

3.1 Access Control List

CAS^2 での Authorization は service パラメータを権限管理の鍵とする認証体系であり，この考え方を “Service Based Authorization” と呼ぶこととする．権限管理は Validation サーブレットによって行われ，アプリケーションから Validation サーブレットに渡された service パラメータを CAS^2 が持つ権限管理データベースと照合する．

CAS^2 において，権限管理データベースは認証データベースと同じく，外部データベースを利用する☆．

権限管理データベースに従って我々は以下の意味での権限管理機構を導入した．

- ST の検証（Validation）要求を受理した際，アクセスを行おうとするユーザが当該の URL に対してアクセス権を持つかどうか．
- アクセス権がある場合にアプリケーションに送信するユーザ情報を任意に設定可能とする．

この権限管理機構を実現するアクセス権限リストを “CAS Access Control List”（CAS-ACL）と呼ぶこととする．CAS-ACL は，LDAP を利用した場合には，以下のように記述される．

☆ 以下では，権限管理データベースは LDAP ディレクトリサーバ内に格納されていると仮定して議論を進める．

- 標準的なCAS-ACLエントリ：
  dn: cn=uPortal,ou=cas,o=NU
  cas-auth-type: basic
  cas-attributes: uid,username,MailAddress
  cas-service: https://app\.foo/.*

各CAS-ACLエントリはアクセス条件とアプリケーションに渡すユーザの属性値が同一であるサービス（URL）からなるクラスを定義する．これを“CAS Access Control Class”（CAS-ACC）と呼ぶこととする．CAS-ACCは，CASを利用するアプリケーションの制限とそのアプリケーションを利用できるユーザを制限する機構を以下のように提供する．

- ユーザ認証の結果得られたユーザ情報とCAS-ACLのcas-allow属性に基づき，「誰が」・「いつ」・「どこから」アクセスできるかを規定する．
- CAS-ACLのcas-service属性に基づき，「どのアプリケーション」にアクセス可能かを規定する．

CAS-ACLによる権限管理の手順は以下のとおりである（詳細は3.3節を参照）．

(1) serviceパラメータに従って，それと一致するcas-service属性値を持つCAS-ACCを選択する（「どのアプリケーション」にアクセス可能か）．
(2) 選択されたCAS-ACCを用いてアクセス権限の検証を行う（「誰が」・「いつ」・「どこから」アクセス可能か）．
(3) アプリケーションへ送信するユーザ情報の属性値をcas-attributesエントリから読み取る．

3.2 Service Ticketの発行方法の変更

我々は権限管理機構の導入と同時に，STの発行方法も変更した．STはアプリケーションへのアクセスのための“One-Time Ticket”の役割を果たすため，いったん利用したSTは再度利用されることはない．したがって，同一ページまたは他のページへのアクセス時には必ずLoginサーブレットへのリダイレクションが発生する．具体的には，1つのページへのアクセスのためには3回のブラウザとCASサーバまたはアプリケーションとの通信が必要となる．

このアクセス回数を減少させるため，我々はValidationサーブレットからアプリケーション経由でブラウザに「次回，同一のCAS-ACCに属するURLにアクセスする際に有効なチケット」を発行するように変更した．このサービスチケット（ST）を“nextticket”と呼ぶこととする．これによって，同一のCAS-ACCに属するURLへのアクセスに際しての通信回数は1回に減少することとなる．また，各CAS-ACCに対してnextticketを発行するか否かは，CAS-ACLのcas-attributesの属性値で指定可能とした．次節（3.3節）ではnextticketに関するセキュリティについても議論する．

3.3 認証メカニズム

アクセス権限機構とnextticketの導入により，従来のCASと比較して，認証メカニズムに多少の違いが生じる．

3.3.1 CAS^{2}認証（1）

まず，2.2.1項と同じく，CAS認証を必要とするアプリケーションにユーザが初めてアクセスする際の動作過程を2.2.1項と比較しながら解説する．

(1) アプリケーション（https://app.foo/）にSTなしにアクセスする．このとき，アプリケーション（CASクライアントライブラリ）からLoginサーブレットへのリダイレクションによって「ログイン画面」が提示される．この過程は2.2.1項(1)と同一である．
(2) 2.2.1項(2)と同じく，CASサーバ（Login）は認証を行い，ブラウザに対してTGCを発行する．ここで，STを発行する際にserviceパラメータおよびユーザ情報などを用いてCAS-ACLとの照合を行う（**図4-1**）．CAS-ACLを用いたアクセス権限の照合（Authorization）に成功したときのみSTを発行し，リダイレクションを発生する（図4-2）．また，CAS^{2}サーバ内のSTデータベースには該当したCAS-ACC（CAS-ACLのdnの値）が保存される．
(3) 2.2.1項(3)と同じく，STアクセスを受理したアプリケーション（CASクライアント）はValidationサーブレットにSTを送付し（**図5-3**），STの正当性・有効性をチェックする（図5-4）．ここでの正当性にはアクセスされたURLから得られるCAS-ACLに基づくアクセス権限の照合だけでなく，STデータベースに保存されたCAS-ACCと，この段階で得られたCAS-ACCとの照合を行う．これらSTの正当性が確認できた場合に，当該のユーザ情報をアプリケーションに送付する．その際に送付するユーザ情

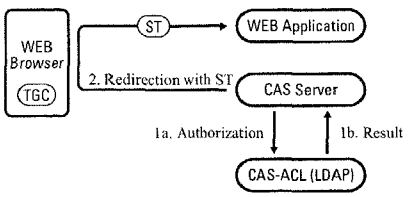

図4 CAS^{2}の動作（1）
Fig. 4 Action of CAS^2(1).

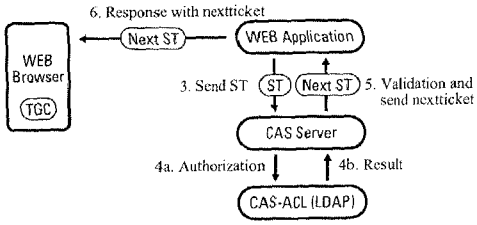

図 5 CAS^2 の動作 (2)
Fig. 5 Action of CAS^2(2).

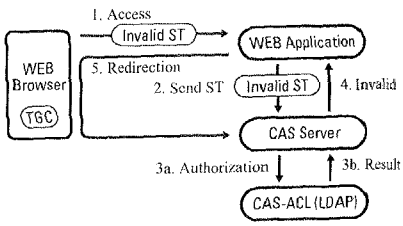

図 6 CAS^2 の動作 (3)
Fig. 6 Action of CAS^2(3).

報は CAS-ACL で指定されたユーザ情報のみである．また，CAS-ACL において nextticket の送付が指定されている場合には，新規に生成した ST をアプリケーション経由でユーザに送付する（図 5-5）．新規 ST は，旧 ST と同一の CAS-ACC に対応するものとして生成する．

以上の過程では，ST を発行する段階と，ST の照合を行う段階の 2 回にわたって “Service Based Authorization” が行われる．重要な検証過程は後者であり，前者で CAS^2 サーバ内部に保存された ST データベースの CAS-ACC 値と後者での照合を行うことにより，service パラメータの改竄による不正アクセス（許可されていない URL へのアクセス：Man-in-Middle attack）を防止することができる．

3.3.2 CAS^2 認証 (2)

次に，nextticket として発行された ST を持つブラウザが前回アクセスした URL と同一の CAS-ACC に属する URL へアクセスする場合を考えてみる．

この場合にはブラウザからアプリケーションへのアクセスにおいて有効な ST が含まれているため，2.2.2 項で示した Login サーブレットへのアクセスは発生せず，アプリケーション（CAS クライアントライブラリ）から Validation サーブレットへのアクセスで ST の有効性が検証される．このアクセス方式では，ブラウザと CAS サーバまたはアプリケーション間の通信回数は 1 回に減少していることが分かる．

一方，異なった CAS-ACC に属する URL へのアクセスを行う場合（図 6-1）には，nextticket として入手した新規 ST の検証の際に，ST 発行時点での CAS-ACC と，アクセスされた CAS-ACC が異なるため，ST の検証が失敗する（図 6-2, 3, 4）．したがって，ST の検証に失敗したアプリケーション（CAS クライアント）は Login サーブレットへのリダイレクションを生成し（図 6-5），ブラウザに対して，新規 ST の取得を求める．しかし，この場合においても TGC 自身が有効である限りブラウザに対して「ログイン画面」の提示は行われないため，従来の CAS 認証過程と同様にシングルサインオン環境は維持されている．

3.4 その他の改良

CAS-ACL に基づく Service Based Authorization のほかに，我々は以下のような機能を CAS^2 で実現した．

- POST メソッドへの対応および国際化対応．
- 二重ログインの防止．
- ユーザを一意に識別可能な任意のキーによるログイン機能．

前節の CAS^2 認証の過程は従来の CAS クライアントで実現可能である．しかしながら，上記の POST メソッドへの対応および国際化は，CAS サーバの改良および CAS で利用されていたリダイレクトのための Java Script の改良が必要となる．そのため，我々はブラウザおよびアプリケーションと CAS^2 サーバ間で通信されるパラメータに以下の事項を追加した．

- CASREQUESTMETHOD パラメータ：アプリケーションへのリクエストメソッドが GET または POST のいずれかを示すパラメータ．デフォールト値は GET とした．これにより，リダイレクトに用いる Java Script を切り替える．
- ENCODING パラメータ：アプリケーションが要求している character encoding の値を示すパラメータ．デフォールト値は UTF-8 とした．これにより，CAS^2 サーバ内部でのパラメータ値のエンコーディングおよび，リダイレクトに用いる Java Script を含む JSP の character encoding を変更する．

これらのパラメータを利用することにより入力メソッドの適切な選択と国際化に対応した．

なお，「二重ログインの防止」は，TGC の新規発行時に既存の TGC を検索し，同一ユーザによる異なるブラウザからのログイン状況を把握することで実現した．

3.5 Access Control List の管理形態

CAS-ACL は CAS^2 の起動時に読み込まれるだけでなく，CAS^2 の動作を妨げることなく随時更新可能である．この機能を実現するために，我々は CAS^2 に

新規のサーブレット（Admin）を導入した．Adminサーブレットにアクセスすることにより，CAS^{2}内部のCAS-ACLデータベースの更新が可能となる．

また，Adminサーブレット自身もCAS-ACLによる権限管理を受けるが，cas-auth-type: trustedで示される，通常とは異なるCAS-ACLの制御に従う．

- trustedなCAS-ACLエントリ（1）：
  dn: ou=cas,o=NU
  cas-allow: (uid=naito)
  cas-auth-type: trusted

このCAS-ACLはLDAP DIT内においてCAS-ACLに関するサブツリーのルートノードであり，ここに示されたuid=naitoに一致するユーザはCAS-ACLの全エントリをアップデートする権限を持つ．

さらに，Adminサーブレットは，CAS-ACLの分散管理も可能な仕様を持つ．

- trustedなCAS-ACLエントリ（2）：
  dn: ou=uPortal,ou=cas,o=NU
  cas-allow: (|(uid=kajita)(uid=naito))
  cas-auth-type: trusted

上に示したCAS-ACLは，ou=uPortal,ou=cas,o=NU以下のサブツリーのルートノードであり，ここに示されたuid=kajitaまたはuid=naitoに一致するユーザはこのサブツリー以下の全エントリをアップデートする権限を持つ．このように，Adminサーブレットによる CAS-ACLの管理機構は，CAS^{2}を利用するアプリケーションの管理者たちによる分散管理を実現している．

4. 運用実績

ここでは，CAS^{2}を用いた実運用結果として，名古屋大学ポータルおよび学務情報システムを用いて行った，平成16年度後期成績入力・平成17年度前期履修登録の実運用結果を報告する．

4.1 名古屋大学ポータルの概要

名古屋大学ポータルは，Webサーバ群（Sun Microsystems社 Sun Fire V210：5台），CASおよびLDAPサーバ（Sun Fire V480：1台），データベースサーバ群（Sun Fire V240：1台およびStorEdge 3150FC：1台）から構成され，Webサーバ群の負荷分散にはNortel Networks社のAlteonを，データベースサーバ群の負荷分散にはOracle 10g Real Application Clusterを利用したシステムで，この上で，大学ポータルフレームワークであるuPortalを用いて名古屋大学ポータルを構築している．

Webでの履修登録および成績入力には，Oracle Application Server（Sun Fire V210：2台およびSun Fire V120：2台）を名古屋大学ポータルと同一の負荷分散装置に接続し，Oracle 9i上のPL/SQLで独自開発したソフトウェアをデータベースサーバ（Sun Fire V240：1台）で利用した．

表1 限界性能試験結果
Table 1 Results of performance test.

処理内容	スループット（ページ/秒）	レスポンス（秒）
履修登録	37.5	1.5
集中登録	60.0	1.1
ポータル	17.5	2.4

CAS^{2}認証については，uPortalではuPortalパッケージおよびJava用CASクライアントを，教務システムについてはPL/SQL用CASクライアントを用いて個別に認証を行った．

4.2 実運用結果

名古屋大学における平成16年度後期成績入力は，授業担当教員約1,000名，科目数約4,000を対象に，2005年2月中旬に19日間（464時間）で行い，アクセス元を名古屋大学学内からに限るという制限を設けて運用した．また，平成17年度前期履修登録は，新2年生から新4年生，約6,500名を対象に，2005年3月下旬に9日間（203時間）で行い，学内だけでなく，学外からのアクセスも許可して運用した．ともに，各日2時間程度の保守時間を設定した以外には，深夜のアクセスも許可した．

また，CAS^{2}の「ログイン機能」として「電子メールアドレス」をログインIDとする認証機能を提供して運用を行った（3.4節参照）．

4.2.1 限界性能試験

履修登録に先立ち，e-Testを用いた負荷実験を行った．**表1**には，ボトルネックとなることがあらかじめ分かったアプリケーション固有のデータベースサーバのCPU使用率が85%以上に達したときの値を示してある．また，CASサーバのサーブレット呼び出し回数も同時に計測し，集中登録時に最大で3,000回/分☆の呼び出しが行われていることを確認した．この限界性能試験においてのCASおよびLDAPサーバのCPU使用率は40〜50%程度で推移した．

この結果から分かるように，レスポンスは最大でも1〜2秒程度で推移しており，外部認証システムであるCAS^{2}認証を用いても高負荷が予想される履修登録において十分な性能が得られることが分かった．

4.2.2 実運用時でのCAS^{2}の性能

成績入力・履修登録の実運用時のCASサーバのサー

☆ データベースサーバの最大負荷とは異なる時点での値である．

表 2 CAS サーブレット呼び出し回数
Table 2 Counts of calling of CAS servlets.

運用内容	最大（回/分）	時間平均（回/時間）
成績入力	500	341
履修登録	1,236	6,986

ブレット呼び出し回数は**表 2** のとおりであり，限界性能試験時の 3,000 回/分の実績と比較して大幅な余裕があったことが分かる．また，履修登録時における CAS サーバのサーブレット反応時間についても計測を行った結果，Login サーブレットについては，平均 0.2 秒程度，Validation サーブレットについては，平均 0.05 秒程度で推移し，アクセス数の増大にともなって急激に反応時間が低下するなどの現象は見られなかった．

5. まとめと今後の課題

本論文では，多様な管理形態を持つ情報システムの統一的かつセキュアな認証基盤の実現の 1 つの方法として，我々が拡張した CAS^2 を用いて構築した認証基盤について論じた．我々が行った CAS への権限管理機構の組み込みは，Queens' University などでも検討されており，Central Authentication and Authorization Service の必要性の存在が明らかであると考えられる．

今回の CAS^2 拡張は CAS Version 2 を基本として行ったが，冒頭にも述べたとおり，新たに発表された CAS Version 3 では Spring Framework による記述に全面的に変更され，種々の機能追加・動作方法の変更などが容易に実現できるようになった．したがって，我々の CAS^2 拡張も Spring Framework 内で実現することにより，CAS の拡張手段の 1 つとして独自に提供でき，CAS 本体との明確な分離により保守性も高まる．これにより，より容易に高機能な Central Authentication and Authorization Service の実現が可能であると考えられる．

また，今回の CAS^2 拡張で見送った機能拡張の例として次のようなものがあげられる．

- CAS サーバ自身の負荷分散
  今回の実運用よりもさらに高負荷環境で CAS を利用するためには CAS サーバ自身を負荷分散対象とする必要があるであろう．そのためには，TGC/ST を保持するデータベースを外部に持ち出すか，Java RMI などのリモート呼び出し機構などを用いて TGC/ST の共有を行うなどの方法が考えられる．
- 広域認証網での CAS の利用
  大学においては，他大学からの訪問者に対してサービスを提供しなければならない場面も少なからずありうる．そのような場面においては他大学などで動作している CAS サーバを経由して認証結果などの交換が必要となる．また，大学間での Authentication および Authorization サービスの実現を目的としている Shibboleth[5] との比較検討も必要である．

これらの機能の実現により，より大規模かつ広範囲な統一認証基盤の実現が可能と考えられる．

謝辞 本研究は，文部科学省平成 16 年度「知的資産の電子的な保存・活用を支援するソフトウェア技術基盤の構築」研究開発課題「ユビキタス環境下での高等教育機関向けコース管理システム」（研究代表者：間瀬健二），および，文部科学省科学研究費基盤研究（A）「地域学術コンソーシアムにおける e-Learning 地域ハブに関する研究」（研究代表者：梶田将司，課題番号：15200054）の助成を受けて実施されている．また，本論文を執筆するにあたり，図版の作成を快諾していただいた名古屋大学大学院多元数理科学研究科・久保仁講師に感謝する．

参 考 文 献

1) Yale University ITS Technology & Planning. http://tp.its.yale.edu/
2) CAS Generic Handler. http://esup-casgeneric.sourceforge.net/
3) JA-SIG. http://www.ja-sig.org/
4) Central Authentication Service. http://jasigch.princeton.edu:9000/display/CAS.
5) Internet2 Working Group, Shibboleth Project. http://shibboleth.internet2.edu/
6) 梶田将司，内藤久資，小尻智子，平野 靖，間瀬健二：CAS によるセキュアな全学認証基盤の構築，情報処理学会研究報告，Vol.2005, No.39, pp.35–40 (2005).
7) 梶田将司，内藤久資，小尻智子，平野 靖，間瀬健二：CAS によるセキュアな全学認証基盤による名古屋大学ポータルの運用，第 3 回 WebCT Conference 予稿集，pp.115–120 (2005).
8) 内藤久資，梶田将司：Central Authentication and Authorization Service — Web Application のための新しい認証システムの試み，京都大学数理解析研究所講究録，Vol.1446, pp.14–39 (2005).

(平成 17 年 7 月 8 日受付)
(平成 17 年 10 月 11 日採録)

内藤 久資

1961年生．1985年大阪大学理学部数学科卒業．1987年名古屋大学大学院理学研究科数学専攻博士課程（前期課程）修了．1988年10月同博士課程（後期課程）中途退学．同年11月名古屋大学理学部助手．1993年より名古屋大学大学院多元数理科学研究科助教授．この間，1989年大阪大学理学部湯川奨学生，1989～1990年英国Warwick大学数学研究所にて研究．微分幾何学，非線型偏微分方程式，コンピュータネットワークに関する研究および大学内・研究科内のコンピュータネットワークに関する業務に従事．理学博士．日本数学会会員．

梶田 将司（正会員）

1990年名古屋大学工学部情報工学科卒業．1995年同大学大学院工学研究科情報工学専攻博士課程満了，名古屋大学工学部助手．1998年同情報メディア教育センター助手．2002年情報連携基盤センター助教授，文部科学省メディア教育開発センター客員助教授併任，2003年株式会社エミットジャパン取締役兼任．博士（工学）．大学における教育・研究活動でのIT活用に関する研究に従事．1996年電気関係学会東海支部連合大会奨励賞，1998年日本音響学会第15回粟屋潔学術奨励賞，2001年電子情報通信学会第56回論文賞．電子情報通信学会，日本音響学会，日本教育工学会，IEEE各会員．

小尻 智子（正会員）

1975年生．2003年名古屋大学大学院工学研究科情報工学専攻博士課程後期課程修了．工学博士．同年名古屋大学大学院情報科学研究科助手，2004年同大学情報連携基盤センター助手，現在に至る．教育支援システム，特に分散環境下での協調学習支援に関する研究に従事．人工知能学会，教育システム情報学会各会員．

平野 靖

1995年名古屋大学工学部電子情報工学科卒業．1997年同大学大学院博士課程電子情報工学専攻前期課程修了．1999年同後期課程修了．2000年同大学大学院工学研究科助手．2002年同大学情報連携基盤センター助手．2004年同助教授．博士（工学）．1998年4月より1999年11月まで日本学術振興会特別研究員（DC2），同年12月より2000年3月まで日本学術振興会特別研究員（PD），3次元画像処理とその肺腫瘤の良悪性鑑別への応用に関する研究，および大学内・大学間の認証システムに関する業務に従事，電子情報通信学会学会，日本医用画像工学会，日本生体医工学会，コンピュータ支援画像診断学会，およびIEEE各会員．

間瀬 健二（正会員）

1979年名古屋大学工学部電気工学科卒業．1981年同大学大学院工学研究科情報工学専攻修士課程修了．同年日本電信電話公社（現NTT）入社．1988～1989年米国MITメディア研究所客員研究員．1995～2002年（株）国際電気通信基礎技術研究所研究室長．2002年より，名古屋大学情報連携基盤センター教授．コンピュータによるコミュニケーション支援の研究を推進している．人工知能学会1999年度論文賞．IEEE，ACM，電子情報通信学会，VR学会，画像電子学会各会員．博士（工学）．