Vol.2010-IOT-8 No.38
2010/3/2

UPKI認証連携基盤を用いたWebアクセス制御

高田 真吾[†1] 金子 直矢[†1] 齋藤 剛[†1]
佐藤 聡[†1] 新城 靖[†1]
中井 央[†2] 板野 肯三[†1]

図書館のような施設に設置される Web 閲覧端末において柔軟なアクセス制御システムを実現するため，HTTP プロクシを利用したシステムを構築した．認証には Shibboleth 認証を用い，それにより取得できる属性に応じて適用するフィルタルールセットを柔軟に設定可能とした．これにより，たとえば提携先機関の図書館などで，自らの所属機関によるアカウント認証を利用できるようになり，従来必要であった一時的アカウントの発行といった煩雑な管理作業を省略することを可能とした．提案システムが処理可能な秒間リクエスト数を計測する実験を行ったところ，通常の Squid プロクシを利用する場合に比べ，無視できるレベルであることを確認した．

Web access control using UPKI Federation

Shingo TAKADA,[†1] Naoya KANEKO,[†1] Go SAITO,[†1]
Akira SATO,[†1] Yasushi SHINJO,[†1] Hisashi NAKAI[†2]
and Kozo ITANO[†1]

This paper describes a flexible access control system for Web that is used in public places including libraries. In this system, visitors are authenticated by Shibboleth, a Single Sign On Protocol. Administrators can describe access control policies using URL patterns in regular expressions and the attributes from Shibboleth. Before visitors access web servers through a proxy server, they are authenticated by Shibboleth servers of their organizations. According to the attributes from Shibboleth, access control rules in the proxy server are dynamically modified. As a result, administrators no longer have to create temporary accounts for visitors. The overhead of the access control system was negligible.

1. はじめに

近年，インターネット接続環境の普及により，調べ物に Web 上の情報を利用する機会が増えてきている．それに伴い，図書館における Web 閲覧端末の需要も増えてきている．その機関の構成員が利用する場合には，アカウントという形で利用者を特定できるため，利用者認証を行った上で Web の利用を許可していることが多い．

具体例としては，筑波大学附属図書館に設置されている端末では，インターネット (学外のサーバ) への接続の際に，筑波大学の有効なアカウントが必要とされている．そのため，他大学の構成員など，学外の利用者がその端末から学外のサイトを利用したい場合，一時的なアカウントを作成してもらい，使用終了後にはそのアカウントを管理者が削除する必要がある．このような一時的アカウントに関する作業はシステムの管理者しか行えず，運用コストが非常に高くなるという問題がある．

この問題を解決する方法として，その利用者が所属する組織で認証を行い，その認証結果をもとに図書館側でアクセス制御を設定するという方法がある．筑波大学附属図書館を訪れた A 大学の学生が，筑波大学の認証サーバではなく A 大学の認証サーバで認証を行い，その結果に応じて附属図書館の端末を使用して Web を利用する権限を発行する，というものである．このような方法を実現するための認証技術として，Shibboleth[1)] がある．日本では，国立情報学研究所の UPKI イニシアティブが中心となり，Shibboleth を利用した認証連携基盤として「学術認証フェデレーション」が構築・運用されている[3)]．

本研究では，Shibboleth 認証の結果に応じて，別に定義するルールセットに基づきプロクシサーバにおける利用者へのアクセス制御をコントロールすることにより，柔軟なアクセス制御の仕組みを実現可能とする．

2. 関連研究

Opengate[5)] は，佐賀大学が開発しているネットワーク利用者認証システムである．江藤らは，この Opengate の認証方法として Shibboleth 認証を実装している[7)]．Opengate は利用者のネットワークへの接続を制御する仕組みであり，L4 レベル，すなわちプロトコル・

†1 筑波大学 システム情報工学研究科
University of Tsukuba, Graduate School of Systems and Information Engineering.
†2 筑波大学 図書館情報メディア研究科
University of Tsukuba, Graduate School of Library, Information and Media Studies.

IP アドレス・ポートといったレベルでのアクセス制御を行っている．

また，大谷らは Opengate の付加機能として，大学ポータルサイトの強制表示機能を実装している[6]．これは，Web を利用する際に認証を必要とし，認証後には強制的にポータルサイトを表示させるというものである．この機能は，Web アクセスに使用される TCP ポート 80 番の通信を監視し，書きかえを行うという手法を用いて実現されている．この手法も，トランスポート層 (L4) レベルでの手法である．

広島大学のキャンパスネットワークである HINET2007 においても，同様の機能が実装されている[4]．HINET2007 では，ネットワークスイッチの認証機能とリダイレクト機能を利用し，Shibboleth 認証を用いた利用者のシングルサインオンを実現している．この手法もまた，トランスポート層 (L4) レベルでの手法である．

これらに対し本研究では，制御対象を Web アクセスに限定し，アプリケーション層 (L7) レベルの制御を行う点で異なる．Web アクセス制御に特化することで，例えば掲示板サイトは閲覧 (GET リクエスト) のみ許可する，URL として特定の文字列を含むサイトへのアクセスを拒否するといったような，より詳細な制御を実現可能としている．

3. 設　　計

本章では，Shibboleth 認証の概要と，Shibboleth 認証を用いた Web アクセス制御の方式について述べる．

3.1 Shibboleth

Shibboleth とは，Internet2 による SAML(Security Assertion Markup Language) を用いたシングルサインオンプロトコルである．

Shibboleth を用いたシステムは，認証サーバである IdP(Identity Provider)，その認証結果に応じたサービスを提供する SP(Service Provider) と，どの IdP を用いて認証を行うか選択させる DS(Discovery Service) から構成される．

ある SP がある組織に所属する利用者を認証したいとき，SP が直接利用者を認証するのではなく，その組織の IdP を用いて認証を行う．その際，SP は利用者を DS に誘導し，DS は利用者に対して「どの IdP で認証を行うか」を選択させ，その IdP に誘導する．利用者は選択した IdP のページで，自分の認証情報を入力することで認証を受ける．その認証結果は，複数の組織が所属して構成されるフェデレーションの内部で利用できるため，一度利用者が自分の選択した IdP で認証を行えば，毎回認証を行うことなくそのフェデレーションに属するサーバのサービスを受けること (シングルサインオンの実現) が可能となる．

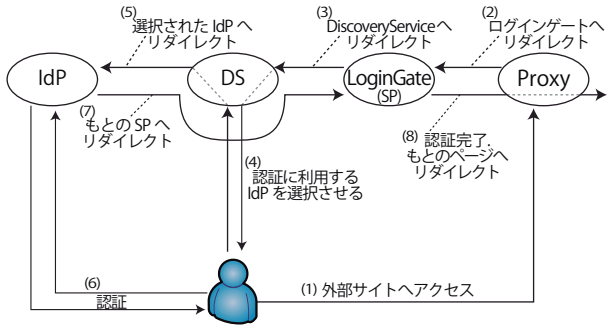

図 1　認証処理の流れ
Fig. 1　Authentication Flow

3.2 HTTP プロクシサーバによる Web アクセス制御

提案システムでは，利用者の認証に Shibboleth を利用する．利用者に対する Web アクセス制御は，HTTP プロクシサーバである Squid[2] の機能を用いることにより実現する．

HTTP プロクシサーバでは，クライアント (Web ブラウザ) からのリクエストを受け，そのリクエストを対象の Web サーバに転送し，その応答をクライアントに返す．Squid では，`url_rewrite_program` という機能によりそのリクエストを加工することできる．この機能を用いることで，クライアントからの Web アクセスをアクセス制御ルールに基づいて処理することを可能とする．

提案システムにおいては，利用者は認証前・認証後のどちらかの状態を持つ．認証前の状態では，明示的に許可されたサイトを除き，あらゆるサイトへのアクセスはプロクシサーバによりログインゲートページへとリダイレクトされる．

ログインゲートページは Shibboleth の SP として動作する CGI スクリプトであり，Shibboleth の認証情報をチェックする．リクエストに利用者の Shibboleth 認証情報が含まれていない場合，DS にリダイレクトし利用者に認証を要求する．利用者は DS で自分が使用する IdP を選択し，その IdP で適宜認証を行う．IdP は認証完了後にログインゲートページにリダイレクトする．ログインゲートページは Shibboleth 認証情報をチェックし，正当な利用権限を持っていると確認できた場合には，あらかじめ設定されたルールに基づいてプロクシサーバのルールを設定する (**図 1**)．利用終了後は，ログアウトページにアクセスすることで認証状態とプロクシサーバのルールが解除される．

Vol.2010-IOT-8 No.38
2010/3/2

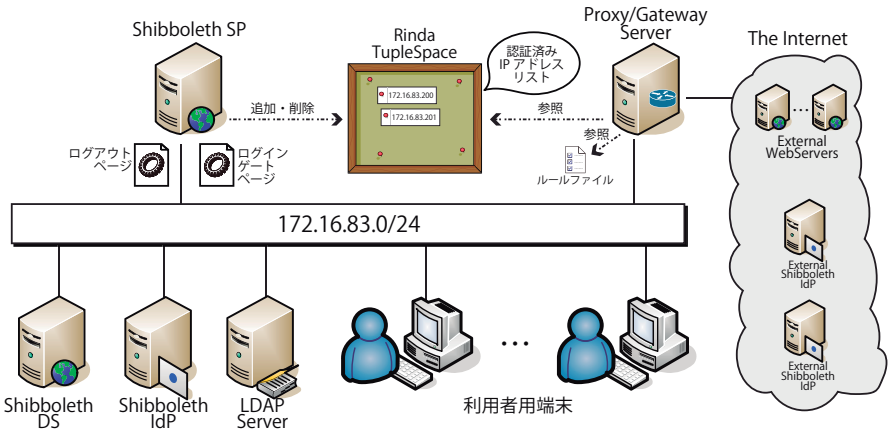

図 2 システム構成図
Fig. 2 System overview

3.3 ルール記述

提案システムでは，利用者の属性に対してルールを記述する．Shibboleth で定義されている属性のうち，所属する組織における職種情報として `eduPersonScopedAffiliation` 属性を，アプリケーションの利用資格情報として `eduPersonEntitlement` 属性を，それぞれ利用する．これらの組み合わせについて，Web アクセスルール群を設定可能とする．

Web アクセスルール群は 0 個以上の Web アクセスルールから構成され，Web アクセスルールは次のような要素を持つ．

url マッチ対象の URL を正規表現で表す

method マッチ対象の HTTP リクエストのメソッドを表す

action マッチしたリクエストを許可 (`ACCEPT`) するか，または拒否 (`REJECT`) するかを表す

4. 実　　装

本章では，提案システムの構成とその実装について述べる．

4.1 システムの構成

提案システムの構成図を**図 2** に示す．クライアントの認証状態の保持には，並列プログラミング言語 (協調言語) である Linda を用いる．特に，今回は実装に Ruby を利用するため，Linda の Ruby 実装である Rinda の TupleSpace を利用する．この TupleSpace を保持するため，専用のデーモンを動作させる．TupleSpace へのアクセスは，Ruby 1.8.5 標準の dRuby を用いる．

認証状態を記録する TupleSpace にその IP アドレスが記録されていないクライアントからのアクセスは，未認証状態のクライアントからの Web アクセスとして扱われ，プロクシサーバのルールによりログインゲートページへ強制的にリダイレクトされる．ログインゲートページでは Shibboleth 認証情報が確認され，認証情報を持たない場合には DS へとリダイレクトされる．正しい認証情報を持つ場合には，ログインゲートにより，そのクライアントの IP アドレスとともに利用者の `eduPersonScopedAffiliation` と `eduPersonEntitlement` が TupleSpace に記録され，利用者が最初にアクセスしようとした URL にリダイレクトされる．

プロクシサーバにおいて，TupleSpace に IP アドレスが記録されているクライアントからのアクセスの場合，その IP アドレスに紐付けられた Web アクセスルールが適用される．チェックの結果アクセスが拒否されれば利用者にその旨通知され，許可されればそのリクエストは許可される．

利用者がログアウトページにアクセスすると，TupleSpace からそのクライアントの IP アドレスが削除され，また Shibboleth の認証情報であるクッキーも削除される．これにより，クライアントは完全に未認証状態に戻る．

4.2 ログインゲート・ログアウトページ

ログインゲートページは，Shibboleth の SP として動作する CGI プログラムである．クライアントからのリクエストを受けると，Cookie に含まれる Shibboleth の認証情報をチェックする．認証情報がない場合には，DS へリダイレクトする．認証情報がある場合には，Shibboleth の属性として `eduPersonEntitlement` と `eduPersonScopedAffiliation` を取得し，TupleSpace に IP アドレスとともに記録する．これにより，このクライアントは以後認証済みとして扱われる．

ログアウトページは，通常の CGI プログラムである．クライアントからのリクエストがあると，その IP アドレスを TupleSpace から削除する．その後，Shibboleth のクッキーを削除し，ログアウトした旨を利用者に通知する．

提案システムでは，Web サーバとして Apache 2.2.3[⋆1]を，Shibboleth SP として shibboleth-2.2 を利用した．CGI スクリプトは Ruby 1.8.5 を用いて記述した．

⋆1 http://www.apache.org/

4.3 URL Rewrite Program

提案システムでは，Squid の `url_rewrite_program` として作成したスクリプトを指定することにより，ルールに基づいた URL の書きかえを実現している．このスクリプトは，標準入力からリクエスト情報を受け取り，標準出力に実際にアクセスを行う URL を出力するスクリプトである．

スクリプトは，まずリクエスト情報に含まれる IP アドレスが TupleSpace の認証済み IP アドレスリストにあるか確認する．リストにない場合は，ログインゲートページの URL を出力する．リストにある場合は，その IP アドレスをキーとして適用すべき Web アクセスルール群を取得する．

次に，各 Web アクセスルールについて，メソッドと URL についてマッチ処理を行い，マッチした場合には `ACCEPT` または `REJECT` の指定に従い，要求された URL または利用者にアクセスが拒否された旨を通知する CGI の URL を出力する．これらの処理を，プロクシサーバが終了されるまで繰り返す．

これらの処理は Web アクセスのリクエストごとに行われるため，そのたびに TupleSpace を参照していては非常にオーバヘッドが大きいという問題がある．そこで，スクリプト内で認証済み IP アドレスリストとルール群を保持しておき，認証済み IP アドレスリストの更新があった場合には，当該部分だけを更新するという方法を用いる．これは，Rinda の `notify` 機能を利用して実現する．

また，Rinda には Tuple の生存期間 (TTL: Time To Live) という機能がある．これは，明示的に TupleSpace から取り除かなくても，TTL で設定した時間が経過すると自動的に TupleSpace から取り除かれるという機能である．ログインゲートページの CGI が TupleSpace にクライアントの IP アドレスと属性を記録する際にデフォルトで設定された TTL を同時にセットすることにより，一定時間後に再び認証を要求するといったことを容易に実現可能となる．

4.4 Web アクセスルール

Web アクセスルールの定義は，プロクシサーバ上の設定ファイルに記述される．アクセス制御の対象とする Shibboleth 属性について，3.3 節で示したような Web アクセスルール群を記述する．これらのルールは YAML 形式で記述する．

図 3 に「学生には，2 ちゃんねるへのアクセスを一切禁止し，Twitter への投稿も禁止するが，それ以外のアクセスは許可する」というルールの例を示す．条件として，`eduPersonScopedAffiliation` が正規表現`^student@.+`にマッチすることを指定し，対象を学生に限定する．次に，URL として正規表現 `2ch\.net`, `bbspink\.com`, `machi\.to` にマッチした場合にそのアクセスを拒否する (`REJECT`) ことを指定している．メソッド (`method`) は全てのメソッドを対象とするために省略している．続いて，Twitter へのアクセス制御ルールが続く．まず，Twitter へのログイン自体は許可するために，URL が `twitter\.com/sessions` にマッチした場合はアクセスを許可する．その後，`twitter\.com` にマッチする URL への POST リクエストを禁止することで，Twitter への投稿を禁止する．これらにマッチしなかった場合には，デフォルトポリシとして `ACCEPT` が指定されているため，アクセスは許可される．

```
- cond:
    affiliation: "^student@.+"
  rules:
    - url:     "2ch\.net"
      action:  REJECT
    - url:     "bbspink\.com"
      action:  REJECT
    - url:     "machi\.to"
      action:  REJECT
    - url:     "twitter\.com/sessions"
      action:  ACCEPT
    - url:     "twitter\.com"
      method:  POST
      action:  REJECT
  default_policy: ACCEPT
```

図 3 提案システムにおけるルール例
Fig. 3 Rule example

これらのルールは，プロクシサーバの起動時にファイルから読み込まれる．ログインゲートページの CGI により TupleSpace に IP アドレスと属性が記録されると，Rinda の `notify` 機能により各 Rewrite Program に通知される．その際，その属性にマッチするルールが検索され，その IP アドレスをキーとするハッシュにルールが記録される．クライアントからリクエストが行われるたびに，リクエストの IP アドレスをキーとしてルールを取得し，そのルールに対して先頭からマッチング処理が行われる．

クライアントがログアウトページにアクセスしたり，TTL が切れて TupleSpace から認証エントリが削除された場合には，`notify` 機能により Rewrite Program に通知され，認証済み IP アドレスリストからの削除処理が実行される．クライアントの IP アドレスとと

もに，紐付けられたルールも削除されるため，プロクシの書きかえルールも未認証状態に戻る．

提案システムでは，クライアントからのアクセスに対して次のような流れでフィルタが適用される．

(1) **ホワイトリスト**
外部 IdP ページや大学内ポータルサイト，OPAC など事前に定義されたホワイトリストと照合され，マッチした場合には未認証状態であってもアクセスが許可される．

(2) **認証状態チェック**
ログインゲートページで認証された IP アドレスからのアクセスかチェックされ，未認証状態であればログインゲートページへリダイレクトされる．

(3) **アクセス先チェック**
Shibboleth 属性である `eduPersonEntitlement` と `eduPersonScopedAffiliation` に応じて設定されるアクセス制御ルールに対してマッチングが行われ，マッチした場合にはその許可に従う．どのルールにもマッチしなかった場合は，そのルールセットのデフォルトポリシに従う．

5. 実　　験

提案システムの性能を測定するため，提案システムが単位時間あたりに処理可能なリクエスト数の測定を行った．

5.1 実 験 環 境

実験環境を**表 1** に示す．今回は，ハードウェアの都合からプロクシサーバをルーティングを行うゲートウェイサーバ lafrenze 上で動作させた．また，SP と IdP に関しては一台のホスト PC 上で，仮想計算機として動作させた．仮想計算機モニタとしては，Citrix 社の XenServer 5.5 を利用した．

5.2 実 験 方 法

提案システムにおけるオーバヘッドを測定するため，ある HTTP サーバ上のファイル 500 個にアクセスを行い，処理された単位時間あたりのリクエスト数を測定する．なお，本研究で対象としている設置場所では，一切の制限なしに直接 HTTP アクセスを許可するケースはまれであり，通常は何らかの認証の仕組みと連携した HTTP プロクシを利用する．そのため，アクセス制御処理を行わなかった場合と行った場合についての比較を行い，その差を提案システムにおけるオーバヘッドと定義する．その上で，比較のためにプロクシを経由

表 1 実験環境

プロクシ・ゲートウェイサーバ lafrenze	
CPU	Intel Core2Duo E7400 2.80GHz / 2Core
Memory	DDR2-SDRAM 4096MB
HDD	HGST HDP725025GLA380 SATA 250GB 7200rpm
NIC1	Intel PRO/1000 MT Desktop Adopter 1000BASE-T for WAN
NIC2	Realtek RTL8168c 1000BASE-T for LAN
OS	Linux 2.6.18-128.4.1.el5
Proxy Server	squid 2.6.STABLE21-3.el5
仮想計算機ホスト abyss	
CPU	Intel Core2Duo E7400 2.80GHz / 2Core
Memory	DDR2-SDRAM 4096MB
HDD	HGST HDP725025GLA380 SATA 250GB 7200rpm
NIC	Realtek RTL8168c 1000BASE-T for LAN
VMM	Citrix XenServer 5.5.0-15119p
SP サーバ orpheus / IdP サーバ eurydice [共にゲスト OS]	
CPU	1CPU
Memory	1568MB
OS	Linux 2.6.18-128.4.1.el5
実験用 HTTP サーバ laila	
CPU	Intel Core2Quad Q6600 2.40GHz / 4Core
Memory	DDR2-SDRAM 4096MB
HDD	HGST HDT72502 SATA 250GB 7200rpm
NIC	Intel PRO/1000 MT Desktop Adopter 1000BASE-T
OS	Linux 2.6.18-128.2.1.el5PAE
HTTP Server	Apache 2.2.3-22.el5.centos.2
L2 Switch	DELL PowerConnect 2708 1000BASE-T

表 2 実験結果

	平均所要時間 [s]	秒間リクエスト数 [回]
プロクシなし	0.552	906
プロクシあり (URL 書きかえなし)	0.607	824
プロクシあり (URL 書きかえあり)	0.628	796

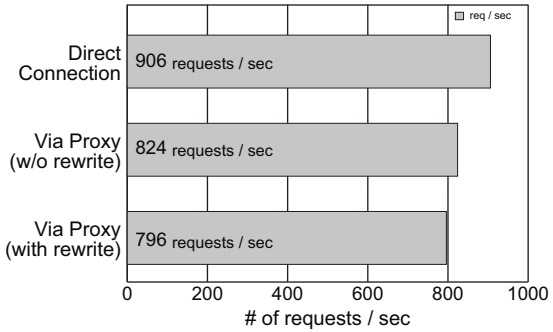

図 4 実験結果
Fig. 4 Benchmark result

しない場合についても同様の測定を行う．アクセス対象は，実験用 HTTP サーバ laila 上に作成された，連番のファイル名を持つ 1024 バイトのファイルとする．この実験は，IdP サーバ eurydice 上で行う．

実験用 HTTP サーバ上の 500 ファイルにアクセスするのにかかった時間を 30 回測定し，その結果から単位時間あたりのリクエスト数を求め，その平均値を測定値として採用する．この測定をプロクシなしの場合，アクセス制御を行わないプロクシを経由する場合，アクセス制御を行うプロクシを経由する場合 (提案手法) の 3 ケースについて行う．この測定を行うスクリプトは Ruby 1.8.5 で作成した．HTTP によるアクセスには，`Net::HTTP` ライブラリを使用し，1 リクエストあたり 1 スレッドを割り当て，可能な限り並列化を行った．

なお，Squid における `url_rewrite_program` のオプションとして，生成するプロセス数を 50，1 プロセスあたりの処理リクエスト数を 50 として実験を行った．また，実験の際に適用するルールは図 3 のものを使用し，Squid のキャッシュ設定は全て無効とした．

5.3 実験結果

実験結果を**表 2** に，グラフを**図 4** に示す．実験結果から，通常の Squid プロクシを経由する場合に比べ，提案手法である URL 書きかえプログラムを経由する場合，秒間リクエスト数は 824 回から 796 回へと減少した．この結果から，提案手法のオーバヘッドは無視できるレベルであると言ってよいことがわかった．

6. おわりに

本稿では，UPKI 認証連携基盤を利用したプロクシサーバによる Web アクセス制御を提案し，手法と実装について述べた．シングルサインオンによる認証の仕組みには Shibboleth を利用した．Web アクセス制御の実現にはプロクシサーバの URL 書きかえ機能を利用し，認証済みの IP アドレスリストの保持には Linda の Ruby 実装である，Rinda による TupleSpace を利用した．実装したシステムによる実験を行った結果，オーバヘッドは十分許容範囲であることを示した．

今後の課題としては，オーバヘッドの削減とより柔軟な Web アクセスルールへの対応があげられる．現在の実装では，利用者が POST リクエストにより送出するデータはアクセス制御の対象とならない．そのため,「特定文字列を含むような掲示板への投稿を禁止する」といったような制御ができない．これを実現するには，POST リクエストにより送出されるデータもアクセス制御の対象とする必要がある．これが実現されれば，提案システムは図書館のような場所に設置される端末を対象とした Web アクセス制御システムとして，非常に柔軟なアクセス制御を実現できると考えている．

Acknowledgement

このプロジェクトは，文部科学省による，組織的な大学院教育改革推進プログラム「ICT ソリューションアーキテクト育成プログラム」の支援をうけて実施された．

参考文献

1) Internet2 middleware architecture committee for education(mace) directory working group. *http://middleware.internet2.edu/dir/ (Accessed: 2010-01-13)*.
2) squid : Optimising web delivery. *http://www.squid-cache.org/ (Accessed: 2010-01-13)*.
3) 学術認証フェデレーション — プロジェクト概要 — shibboleth — upki イニシアティブ. *https://upki-portal.nii.ac.jp/SSO (Accessed: 2010-01-13)*.
4) 藤村喬寿, 西村浩二, 相原玲二. 大規模キャンパスネットワークにおける sso 認証の設計と実装. 電子情報通信学会技術研究報告, Vol. 109, No. 299, pp. 13–18, 2009.
5) 渡辺健次, 只木進一, 江藤博文, 渡辺義明. 利用者認証と利用記録機能を実現するゲートウェイシステム opengate の開発. 電子情報通信学会技術研究報告. IN, 情報ネットワーク, Vol.99, No. 591, pp. 43–48, 2000.
6) 大谷誠, 江藤博文, 渡辺健次ほか. ポータルサイトの強制表示とシングルサインオン. 電子情報通信学会技術研究報告, Vol. 109, No.60, pp. 29–34, 2009.
7) 江藤博文, 大谷誠, 渡辺健次, 只木進一. Opengate とシングルサインオン. 電子情報通信学会技術研究報告. IA, インターネットアーキテクチャ, Vol. 108, No. 459, pp. 259–264, 2009.