情報処理学会論文誌 Vol. 50 No. 8 1856–1869 (Aug. 2009)

CaStor：Web資源に対するケーパビリティの管理・配布を行うWebサーバ

馬渕充啓[†1] 池嶋俊[†1] 川崎仁嗣[†1]
吉野純平[†1] 松井慧悟[†1] 新城靖[†1]
佐藤聡[†2] 上川大介[†1] 加藤和彦[†1]

近年，多くのユーザが，保護されたWeb資源を用いて協調作業を行う場面が増加してきている．これらのWeb資源は，ユーザ間で共有可能なものとユーザ間で共有不可能なものに分類される．既存のWeb資源を用いた協調作業で他のユーザにアクセス権限を委譲して作業分担を行おうとした場合，前者ではユーザ登録をしなければならないという問題があり，後者では既存のWeb資源のプログラムを変更しなければならないという問題がある．本論文では，既存の保護されたWeb資源において，ユーザ間でアクセス権限の委譲を可能にするWebサーバCaStorについて述べる．CaStorは，利用者認証に必要な情報（ユーザIDとパスワード）やWeb資源のURLからケーパビリティを作成し他のユーザに安全に配布する機能を持つ．この機能を利用することで，既存のWeb資源のプログラムを変更することなく権限委譲を行うことができるため作業分担が容易になる．CaStorでは，保存されたケーパビリティを用いてWeb資源にアクセスを行う場合，機密情報を他のユーザに渡す必要がないようにするためにプロキシを用いる．また，このプロキシを用いることでURLのパターンマッチを行いアクセス可能なWeb資源の制限を可能にする．最後に，CaStorを用いて保護されたWeb資源を取得するまでの処理時間の計測を行い，実用に問題がないことを示す．

CaStor: A Web Server for Management and Distribution of Capabilities to Access Web Resources

Mitsuhiro Mabuchi,[†1] Shun Ikejima,[†1]
Satoshi Kawasaki,[†1] Junpei Yoshino,[†1]
Keigo Matsui,[†1] Yasushi Shinjo,[†1] Akira Sato,[†2]
Daisuke Kamikawa[†1] and Kazuhiko Kato[†1]

In recent years, users often work together through protected Web resources. These Web resources are classified into two types: sharable ones and non-sharable ones. In cooperative work using existing Web resources, a user wishes to delegate his or her access rights to other users to assign his or her tasks. However, in the former type, it is necessary to register a user to the Web resources. In the latter type, programs of existing Web resources need to be changed. In this paper, we describe CaStor, a Web server for management and distribution of capabilities to access existing Web resources. CaStor enables a user to delegate access rights to other users. CaStor has two functions: first, to create a capability of a protected Web resource from a user ID, password, and URL, and second, to securely distribute a capability to other users. Using these functions, a user can easily assign his or her tasks by distributing capabilities to other users. In CaStor, a user accesses Web resources by using stored capability through a proxy that enables a user not to pass sensitive information to other users. In addition, the proxy restricts accessible Web resources by pattern matching of URLs. Finally, we measure processing time of getting a Web resource by using CaStor and present practicality of CaStor.

1. はじめに

近年，多くのユーザが，利用者認証とACL（Access Control List）により保護されたWeb資源を用いて協調作業を行う場面が増加してきている．たとえば，文書作成，スケジュール管理，会議室予約，オークションの出品・落札，あるいは，Webショッピングでの買い物等

†1 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba
†2 筑波大学情報環境機構学術情報メディアセンター
Academic Computing & Communications Center, University of Tsukuba

がある．それにともない，1 つの保護された Web 資源で行う作業を複数のユーザで分担したいという要望が大きくなってきている．

保護された Web 資源で行う作業をユーザ間で分担する場合，保護された Web 資源の所有者は他のユーザがその Web 資源にアクセスすることを許可する必要がある．既存の保護された Web 資源は，ユーザ間で共有可能なものとユーザ間で共有不可能なものに分けることができる．前者の場合，ユーザ登録をしていないユーザには，そのユーザに対して ACL を記述することができないためアクセスを許可することができないという問題がある．このため，運用上の問題により組織外の人をユーザ登録できない場合，その人に許可を与えることができず作業分担を行うことができない．たとえば，ある組織内で使用するグループウェアでは，他の組織の人をユーザ登録できないため作業分担を行うことができない．従来，よく用いられる解決方法として，自分のユーザ ID とパスワードを他のユーザに渡すことが行われることがある．しかし，この方法を用いた場合，そのユーザの持っているすべてのアクセス権限を渡すことになってしまうため本来行うべきではない．また，たとえ組織的にはユーザ登録が可能であっても，登録のためには個人情報を提供しなければならないことが多いため，ある作業を受け持つためだけにユーザ登録をしたくないというユーザがいることがある．この場合も，ACL を記述してそのユーザにアクセスを許可することができないため，作業分担を行うことができない．

後者の場合，あるユーザが他のユーザに自分の保持する Web 資源へのアクセスを許可することは考慮されていない．したがって，自分以外のユーザに対してアクセスを許可できないため作業分担を行うことができない．たとえば，文書作成，オークションの出品・落札，そして，Web ショッピングでの買い物等を，1 人で行うことは容易であるが複数人で分担することは実質不可能である．この問題を解決するためには，既存の Web 資源のプログラムを変更する必要がある．

本論文では，上述した問題を解決する，既存の Web 資源に対するケーパビリティの管理・配布を行う Web サーバ CaStor について述べる[15]．ケーパビリティとは，あるオブジェクトへの参照とそのオブジェクトに対して可能な操作を保持するものである[1),5),7),14),17),24)]．CaStor は，保護された Web 資源にアクセスするために必要な情報（ユーザ ID とパスワード）や Web 資源の URL からケーパビリティを作成し他のユーザに安全に配布する機能を持つ．このとき，ケーパビリティには，アクセス可能な URL のパターン，有効期限，そして，使用回数等の制限を付加することができる．これらの機能により，既存の Web 資源のプログラムを変更することなく他のユーザにケーパビリティを渡すことで権限委譲を行うことができるため作業分担が容易になる．CaStor は，単なるパスワード管理 Web サーバではなく，保護された Web 資源にアクセスするために必要な情報からケーパビリティを作成し，それをユーザ間で受け渡すことで作業分担を支援する Web サーバである．

CaStor に保存されたケーパビリティを用いた Web 資源へのアクセスは，機密情報を他のユーザに渡す必要をなくすためにプロキシを通して行われる[13)]．また，このプロキシを用いることで URL のパターンマッチによるアクセス可能な URL の制限を利用することができる．CaStor は，信頼できるコミュニティでのみ使用されると仮定する．信頼できるコミュニティとは，たとえば，家族，友人，そして，仕事関係等である．したがって，CaStor では，ケーパビリティの特徴を生かし作業分担の支援という目的を達成するために，ケーパビリティの利用者や配布の制限は行わない．本論文では，CaStor を用いて保護された Web 資源を取得するまでの処理時間を計測し，実用に問題がないことを示す．

本論文は以下の章で構成される．2 章では，CaStor に対する要求要件とそれに基づいた設計について述べる．3 章では，CaStor の実装について述べる．4 章では，CaStor の実用性を調べる実験について述べる．5 章で本研究の関連研究について述べ，6 章で本論文のまとめを行う．

2. CaStor の設計

2.1 要求要件

CaStor では，既存の保護された Web 資源において，効率良くユーザ間でアクセス権限の委譲を可能にするために以下の要件を満たすように機能を設計する．

(1)　既存の Web 資源のプログラムを変更しないで，アクセス権限の委譲を可能にする．
(2)　Web 資源の所有者でなくても，アクセス権限に対する作業を可能にする．
(3)　既存のアクセス権限からより制限されたアクセス権限の作成を可能にする．

また，CaStor では，自分で作成したアクセス権限や受け取ったアクセス権限の管理を容易にする機能も提供する．

アクセス権限を他のユーザに委譲する機能は，既存の Web 資源においてもプログラムを変更することで実現することができる．しかし，すべての既存の Web 資源のプログラムを変更することは非常に大きな手間がかかる．CaStor では，要件 (1) を満たすためにプロキシを用いてユーザの代わりに既存の保護された Web 資源にアクセスを行う[13)]．プロキシを用いることで，機密情報を他のユーザに渡すことなく他のユーザの ID を使用してその Web 資源にアクセスを行うことを可能にする．また，URL のパターンマッチを行いアクセス可

能な Web 資源を制限することができる．詳しくは，2.8 節で述べる．また，CaStor とプロキシは，同一のホストで動作させ，そのホストは安全であるものとする．

CaStor では，要件 (2) と要件 (3) を満たすために，ケーパビリティに基づくアクセス制御方式を用いる[1),5),7),14),17),24)]．ケーパビリティとは，あるオブジェクトへの参照とそのオブジェクトに対して可能な操作を保持するものである．この方式を用いることで，Web 資源の所有者やシステムの管理者でなくても，ケーパビリティを所持しているユーザであれば誰でもケーパビリティの作成や委譲を行うことができる．そのため，友人に委譲したケーパビリティが，友人の手によって友人の友人に再委譲される可能性がある．このことを，ケーパビリティに似た性質を持つ電話番号と対比して検討する．日常生活において，友人の電話番号を別の友人に教えるということは，非常に慎重に行われる．これは，電話番号は簡単に変えられるものではなく，簡単に他人に教えるものではないという共通の理解が人々の中にあるからである．ケーパビリティも電話番号と同じように，その性質を人々が理解したうえで再委譲は慎重に行われる必要がある．我々は，CaStor は家族，友人，あるいは，仕事関係等の信頼できるコミュニティでのみ使用されると仮定することで，電話番号と同じように再委譲が慎重に行われると考える．したがって，CaStor では，ケーパビリティに基づくアクセス制御の利点を生かすため，他のシステムと同様に，ケーパビリティの利用者制限や配布制限を行うことはしない[1),5),7),14),17),24)]．

CaStor では，ユーザ ID とパスワード等の保護された一群の Web 資源にアクセスするために必要な情報を 1 個のオブジェクトとして扱う．そのオブジェクトのことを本論文では，**Web オブジェクト**と呼ぶ．アクセス権限の委譲に関しては，この Web オブジェクトを渡すことが考えられるが，この方法では他のユーザにユーザ ID とパスワード等の情報に直接アクセスされてしまうという問題がある．そこで，CaStor では，Web オブジェクトの代わりに，それを参照するリンクを配布することにする．CaStor では，このリンクのことを**ケーパビリティ**と呼ぶ．リンクとはいえケーパビリティをそのまま渡した場合，すべての権限を渡すことになるため，CaStor では Web オブジェクトにさまざまな制限を付加することができるようにする．たとえば，有効期限や使用回数制限等である．

要件 (3) に関して，有効期限の延長等により付加された制限情報を編集したい，あるいは，その作業を他人にさせたいという要求や，ケーパビリティへのケーパビリティを作成することで制限を 2 重 3 重に付加したいという要求がある．単純に Web オブジェクトに付加された制限情報を編集可能にした場合，ケーパビリティを持つユーザは誰でもその制限情報を編集することができてしまう．このため，制限情報の編集やその作業を他人にさせたいという要求に対してはよいが，制限を 2 重 3 重に付加したいという要求に対しては問題が残る．そのため，CaStor では，**ハード・ケーパビリティ**と**ソフト・ケーパビリティ**の 2 種類のケーパビリティを用意する．ハード・ケーパビリティとは，Web オブジェクトへのリンクのことである．それに対して，ソフト・ケーパビリティとは，ケーパビリティへのリンクのことである．ソフト・ケーパビリティを作成することで制限を 2 重 3 重に付加することが可能になる．また，ハード・ケーパビリティを用いて制限情報を変更した場合，元の制限情報が変更されるが，ソフト・ケーパビリティを用いて制限情報を変更した場合，そのソフト・ケーパビリティの保持する制限情報が変更される．

ユーザが大量のケーパビリティを保持することにより，ケーパビリティの管理やブラウズが困難になるという問題がある．そのため，CaStor では，ケーパビリティを**ディレクトリ**にまとめて保存する．これにより，ユーザがケーパビリティをある属性に基づいてカテゴライズして保存できるようにする．

配布機能としては，インボックスを用いる方法と直接受け取り手のディレクトリに送る方法が考えられる．CaStor では，メールを送るのと同じように簡単にケーパビリティを送ることができるようにするため，必要でないケーパビリティによってディレクトリ内の必要なケーパビリティが埋もれてしまうという，スパムメールと同じ問題が発生することが考えられる．そのため，CaStor では，**インボックス**を用意し，受け取ったケーパビリティをインボックスに一時的に保存することにする．これにより，必要でないケーパビリティが大量にディレクトリに保存されることを防ぎ，ユーザが必要なケーパビリティのみを選択しディレクトリに保存することができるようにする．

2.2　CaStor のモデル

図 1 に CaStor のモデルを示す．CaStor はケーパビリティの管理・配布を担当し，ケーパビリティを用いた外部の保護された Web 資源へのアクセスにはプロキシ[13)] を用いる．プロキシを用いることで，機密情報を他のユーザに渡す必要をなくすことができる．利用者はブラウザを用いて CaStor にアクセスすることになるが，ブラウザ・CaStor 間とブラウザ・プロキシ間で情報漏洩を防ぐため https でのみ通信可能にしている．

CaStor は，2.1 節で述べたオブジェクト（Web オブジェクト，ディレクトリ，ケーパビリティ，そして，インボックス）を管理する．プロキシは，CaStor から受け取った情報を用いて外部の保護された Web 資源にアクセスを行い，その Web 資源から受け取った応答をそのままブラウザに返す．次節以降，CaStor の扱うオブジェクトについて詳しく述べる．

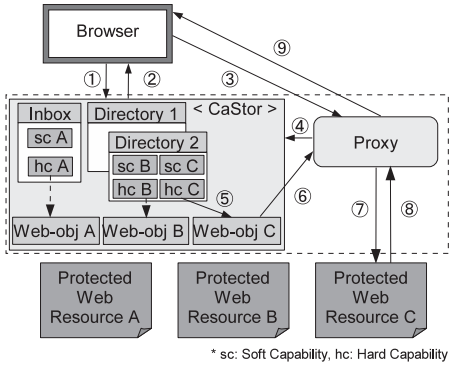

図 1　CaStor のモデル
Fig. 1　The model of CaStor.

2.3　Web オブジェクト

Web オブジェクトとは，外部の保護された一群の Web 資源にアクセスするために必要な URL，ユーザ ID，そして，パスワード等の情報を保持するオブジェクトである．Web オブジェクトは，必要な情報を CaStor に登録することで作成される．その際，その Web オブジェクトを参照するハード・ケーパビリティが作成される．Web オブジェクトは，ハード・ケーパビリティにより直接参照される．これにより，ユーザはケーパビリティを介して Web オブジェクトを使用する．Web オブジェクトは，直接参照しているハード・ケーパビリティがなくなったときに CaStor 内から自動的に削除される．

2.4　ケーパビリティ

ケーパビリティとは，Web オブジェクトや他のケーパビリティへのリンクである．Web オブジェクトではなくケーパビリティを他のユーザに渡すことで，Web オブジェクトの持つユーザ ID やパスワードには直接アクセスさせずに外部の保護された Web 資源にアクセスさせることができる．

ケーパビリティを他のユーザに渡す場合，以下のように制限をかけたいことがある．

- 明日までに作業を終わらせてもらいたいが，それ以降はアクセスを制限したい．
- アンケートへの回答では，回答の回数を 1 回に制限したい．
- Web テストの受験では，期間の制限と回数の制限をしたい．
- 会議室予約を分担したいが，それ以外のスケジュール管理やメールボックス等の資源にアクセスできないように制限したい．

CaStor では，上述した要求を満たすために，ケーパビリティに属性として以下のような制限を設定することができるようにする．

- アクセス可能な URL のパターン．
- ケーパビリティの有効期限や使用回数．

2.5　ソフト・ケーパビリティ

ソフト・ケーパビリティは，元のケーパビリティ（ハード，または，ソフト）をもとに作成される．ソフト・ケーパビリティにも，独自にアクセス可能な URL のパターン，有効期限，そして，使用回数制限を設定することができる．ソフト・ケーパビリティ独自の属性の場合，ソフト・ケーパビリティを保持するユーザは誰でも編集することが可能である．しかし，元のケーパビリティの属性を編集することはできないようになっている．この特性を用いることで，制限を 2 重 3 重に設定することができる．たとえば，有効期限の属性を持つソフト・ケーパビリティをもとにして，使用回数制限の属性を持つソフト・ケーパビリティを作成した場合，このソフト・ケーパビリティは有効期限と使用回数制限という 2 重の制限を持つことになる．また，ソフト・ケーパビリティの制限は元のケーパビリティの制限を超えることはできない（元のよりも有効期限が長い，使用回数が多い等）ため，元のケーパビリティの制限を超えて利用することはできない．

ソフト・ケーパビリティは，あるケーパビリティを選択しそれをもとに作成されるため，ここで循環参照は起こらない．また，循環参照を防ぐため，すでに作成されているソフト・ケーパビリティの参照先を変更することは許していない．

2.3 節で述べた Web オブジェクト，2.4 節で述べたハード・ケーパビリティ，そして，本節で述べたソフトケーパビリティの機能を用いることで 2.1 節で述べた要件 (2) と要件 (3) を満たすことが可能になる．

2.6　ディレクトリ

CaStor では，ユーザは複数のディレクトリを持つことができる．たとえば，プライベート用や仕事用のディレクトリを持つことができる．また，ディレクトリ内に別のディレクトリを作成することで階層的に管理することができる．たとえば，**図 2** のように，プライベート用ディレクトリ内にサークル用，研究室用，そして，授業用ディレクトリを作成することができる．

CaStor では，ユーザが CaStor にログインすることにより，ユーザの保持するディレクトリの 1 つが表示される．ディレクトリ内にケーパビリティが保存されている場合，ケーパビリティの一覧が表示される．図 2 にそのスクリーンショットを示す．また，ディレクト

図 2　ディレクトリを用いたケーパビリティのブラウズ
Fig. 2　Browsing capabilities using a directory.

リ内のケーパビリティのリンク先はプロキシへの URL となっており，ユーザがケーパビリティを選択するとブラウザはプロキシにアクセスを行う．その後，プロキシはケーパビリティを用いて Web オブジェクトにアクセスを行い，必要な情報を取り出す．

2.7　インボックス

CaStor では，インボックスを用いてケーパビリティを配布する機能を提供することで，機密情報を安全に配布することができるようにする．インボックスは，1 ユーザにつき 1 つ用意する．

ケーパビリティの配布を**図 3** に示す．まず，ユーザ A が自分のディレクトリ内から送信したいケーパビリティを選択し配布の手続きを行う．選択されたケーパビリティは，ユーザ B のインボックスに配布される．その後，ユーザ B はインボックス内から必要なケーパビリティを選択して特定のディレクトリに保存する．

CaStor では，ケーパビリティの作成者でなくても保持者であればケーパビリティを配布することが可能であるため権限委譲の際の手間を省くことができる．たとえば，あるプロジェクトを行っている際にリーダがサブ・リーダにある権限を委譲する．従来，その権限を一般メンバに委譲したい場合，リーダが委譲を行うための作業を行うが，CaStor ではサブ・リーダが一般メンバにもその権限を委譲できるためリーダの手間を省くことができる．

2.6 節で述べたディレクトリと本節で述べたインボックスの機能を用いることで，自分で作成したケーパビリティや渡されたケーパビリティの管理を容易にする．

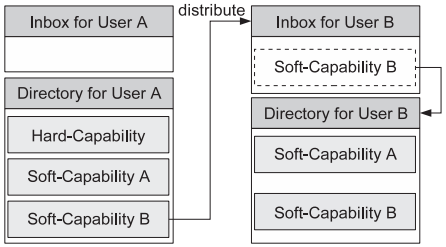

図 3　ケーパビリティの配布
Fig. 3　Distribution of a capability.

2.8　プロキシ

CaStor では，プロキシを用いて外部の保護された Web 資源に間接的にアクセスを行う[13]．このプロキシは参考文献 13) で提案した手法におけるプロキシサーバとして動作する．参考文献 13) で提案したプロキシサーバにおいては，ケーパビリティを持つユーザは，自分が持つケーパビリティを表す URL に基づいてプロキシサーバにアクセスする．プロキシサーバは HTTP により送られてきた URL から取り出した乱数をキーとしてデータベースにアクセスし，保護された Web 資源のオリジナルの URL や機密情報を取得し，それらを用いて Web 資源にアクセスする．

本システムで用いるプロキシは，以下に示す 2 点について，このプロキシサーバを改良したものである．1 点目は，ユーザからのアクセスには，URL だけではなくクッキーを用いるようにしていることである．2 点目は，保護された Web 資源にアクセスする際に，その URL が指定した正規表現に一致するか否かで，アクセスを許可・拒否できる機能を追加していることである．またデータベースへのアクセスを CaStor へのアクセスとなるように変更している．

これにより，このプロキシサーバは，CaStor の一機能として，ブラウザからの Web 資源へのアクセス要求を解釈し与えられた権限の範囲で Web 資源に対してアクセスを行いその応答をそのままブラウザに返す．図 1 を用いて，プロキシを用いた外部の保護された Web 資源へのアクセスの仕組みについて示す．

(1)　ユーザは，ブラウザを用いて CaStor にアクセスし，自分のディレクトリ内のケーパビリティの中から外部の保護された Web 資源 C にアクセスするためのケーパビリティを選択する．

(2)　CaStor は，乱数を含むプロキシへの URL とクッキーをブラウザに返す．
(3)　ブラウザは，受け取った URL とクッキーを用いてプロキシにアクセスする．
(4)　プロキシは，URL に含まれた乱数とクッキーを用いて外部の保護された Web 資源 C にアクセスするために必要な情報を CaStor に要求する．
(5)　CaStor は，乱数とクッキーで確認したケーパビリティを用いて Web オブジェクトから URL，ユーザ ID，そして，パスワード等の情報を取り出しプロキシに情報を返す．
(6)　プロキシは，受け取ったその情報を用いて外部の保護された Web 資源 C にアクセスを行う．
(7)　プロキシは，返ってきた応答（ステータスコード，ヘッダ，メッセージボディ）を受け取る．
(8)　プロキシは，受け取った応答をそのままブラウザに返す．

また，このプロキシを用いることでアクセス可能な URL を制限することができる[13]．たとえば，あるグループウェアで会議室予約（http://example.com/?page=room）とスケジュール管理（http://example.com/?page=schedule）の機能があるとき，前者の URL にはアクセス可能だが後者の URL にはアクセスさせないことができる．この場合，Web オブジェクトとして前者の URL だけを登録するか，アクセス可能な URL のパターン（正規表現）として “?page=room” だけを登録する．プロキシは，Web オブジェクトからの情報を受け取り，その URL を使って Web 資源にアクセスしクライアントにその応答を返す．

本節で述べたプロキシの機能を用いることで，2.1 節で述べた要件 (1) を満たすことが可能になる．

2.9　CaStorMe ブックマークレット

CaStor では，Web オブジェクトを容易に作成するための方法として，CaStorMe というブックマークレットを提供する．ブックマークレットとは，Javascript で書かれ Web ブラウザのブックマークに登録して使用されるプログラムのことである．

CaStorMe を利用するには，ユーザは，登録したい Web サイトの入力フォームにユーザ ID とパスワードを入力し目的のサイトに送信する前に，ブックマークに登録してある CaStorMe をクリックする．すると，CaStorMe は，フォームを解釈し URL，ユーザ ID，そして，パスワード等の必要な情報を取得しウィンドウを表示する．このとき，すでに CaStor 内に作成されている Web オブジェクトに類似のものがある場合，CaStorMe はそれらの一覧を表示することで同一の Web オブジェクトが誤って複数作成されることを防ぐ．一覧に目的のものがない場合，そのウィンドウ内で作成する Web オブジェクトに対応するハード・ケーパビリティの保存先ディレクトリを選択する．その後，CaStorMe は取得した情報を CaStor に送信し，CaStor は受け取った情報を用いて Web オブジェクトを作成し指定されたディレクトリにハード・ケーパビリティを作成する．

2.10　CaStor を用いた問題解決

CaStor を用いることで，ある既存の外部の保護された Web 資源にアクセスするための権限を作成し他のユーザに委譲することで作業分担を行うことができる．以下に，保護された Web 資源を用いた作業分担において想定される問題を示し，その後 CaStor を用いた解決方法を示す．

(1)　Web 上で文書や表を作成しそれらの校正を他のユーザに依頼したい場合，そのシステムに対してそのユーザをユーザ登録しなければならない．
(2)　組織内で運用されているグループウェアに対して組織外のユーザに作業を分担する場合，外部のユーザをそのグループウェアに運用上の理由から登録できない．
(3)　Web ベースのグループウェアで会議室予約を私設秘書に依頼する場合，グループウェアにログイン可能であればそれ以外のスケジュール管理等の操作も可能になる．

このような場合，作業を分担してもらいたいユーザは，CaStor に Web 資源にアクセスするために必要な情報を Web オブジェクトとして登録する．その後，その際に作成されるケーパビリティを外部のユーザに渡すことで権限を委譲し作業分担を行うことができる．また，このとき，Web オブジェクトにその作業で必要な Web 資源の URL のパターンと有効期限や使用回数を設定することで，すべての権限を渡すことなく権限委譲を行うことができる．CaStor はこれらの機能を Web 資源のプログラムを変更することなく追加することができる．

3.　CaStor の実装

この章では，2 章で述べた設計に基づいて行った実装について述べる．CaStor は，Ruby on Rails[18] を用いて実装し，データベースには SQLite を使用している．プロキシは，Java を用いて実装されている[13]．また，2 章で述べたように，CaStor とプロキシは同一のホスト上で動作しそのホストは十分安全であるため，CaStor・プロキシ間の通信は盗聴されることはない．サーバ機材の盗難等の物理的な攻撃に対しては，必要であればハードディスクの暗号化等の既存技術を用いて保護していると仮定する．CaStor とプロキシの間の通信の詳細については，3.3 節で詳しく述べる．

3.1　CaStor のデータ構造

CaStor は，以下の 6 つのデータ構造を持つ．

- User：CaStor に登録されたユーザの情報を保持する．属性としては，user_ID，user_name，password，および，directory_ID 等がある．
- Web_object：保護された Web 資源にアクセスするためのユーザ ID，パスワード，そして，Web 資源の URL 等の情報を保持する．属性としては，web_object_ID，user_ID，passwd，url，参照しているハード・ケーパビリティの数，および，有効期限等の制限情報等がある．
- Directory：ケーパビリティを保持するディレクトの情報を保持する．属性としては，directory_ID，ディレクトリ名，user_ID，および，他の directory_ID 等がある．
- Capability：Web オブジェクトへの参照を保持する．capability_ID，directory_ID，ケーパビリティ名，タイプ（ハード・ケーパビリティかソフト・ケーパビリティかの区別），および，web_object_ID，あるいは，link_data_ID 等がある．
- Link_data：ソフト・ケーパビリティのための制限情報を保持する．属性としては，link_data_ID，リンク先の capability_ID，および，有効期限等の制限情報等がある．
- Inbox：受け取ったケーパビリティを一時的に保持する．属性としては，inbox_ID，user_ID（インボックスを保持するユーザの ID），ケーパビリティ名，send_user_ID（送信元のユーザの ID），および，capability_ID 等がある．

3.2　ケーパビリティの実装

CaStor では，ハード・ケーパビリティとソフト・ケーパビリティの 2 種類のケーパビリティを扱うことができる．**図 4** にハード・ケーパビリティとソフト・ケーパビリティの実装方法を示す．ハード・ケーパビリティとソフト・ケーパビリティは，Capability テーブル内に保持されタイプ属性により区別される．ハード・ケーパビリティA は Web オブジェクトへのポインタとなる web_object_ID を持ち，ソフト・ケーパビリティB，C はリンク・データへのポインタとなる link_data_ID を持つ．ソフト・ケーパビリティB はハード・ケーパビリティA へのソフト・ケーパビリティであるため，リンク・データ B はハード・ケーパビリティA を参照するための capability_ID を持つ．ソフト・ケーパビリティC はソフト・ケーパビリティB へのソフト・ケーパビリティであるため，リンク・データ C はソフト・ケーパビリティB を参照するための capability_ID を持つ．このため，ソフト・ケーパビリティC は，リンク・データ B と Web オブジェクトの両方の属性（有効期限や使用回数制限等）により制限を受ける．このように，ソフト・ケーパビリティへのソフト・ケーパビリティを作成することにより 2 重 3 重に制限を付加することを実現している．

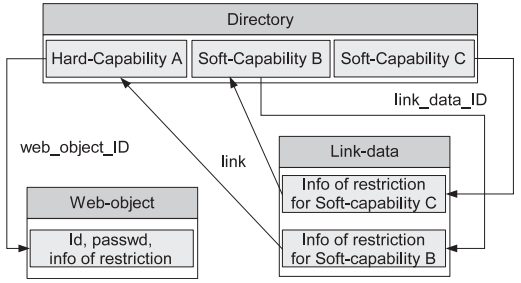

図 4　ハード・ケーパビリティとソフト・ケーパビリティの実装方法
Fig. 4　Implementation of hard-capabilities and soft-capabilities.

有効期限や使用回数等の制限情報の変更を行う場合，ハード・ケーパビリティA では，web_object_ID を用いて Web_object テーブル内に保持された属性情報を変更することができる．それに対して，ソフト・ケーパビリティB，C では，link_data_ID を用いて Link_data テーブル内にそれぞれ保持されたリンク・データ B，C の属性情報を変更することができる．これにより，ソフト・ケーパビリティでは，元のケーパビリティの属性情報を変更できないようにしている．

3.3　一時的オブジェクト

外部の保護された Web 資源にアクセスする場合，CaStor はプロキシへの URL をブラウザに返す．この URL をブックマークに登録する等の方法を用いて，利用者が CaStor からログアウトした後に使用した場合，CaStor でケーパビリティの使用回数や有効期限等の制限情報を検証することができないという問題がある．この問題を解決するために，CaStor では，**一時的オブジェクト（Transient Object）**を用いる．

ユーザがケーパビリティを使用して保護された Web 資源にアクセスする際，CaStor はケーパビリティごとに一時的オブジェクトを作成する．一時的オブジェクトとは，乱数を含んだプロキシへの URL とケーパビリティの識別子を持つものである．この乱数は，他のユーザに簡単には類推されないぐらい大きいものとする．乱数の代わりにケーパビリティの識別子を URL に含めた場合，この識別子は非常に類推しやすいものであるため安全とはいえない．そのため，CaStor では，識別子そのものではなく乱数を URL に含めている．一時的オブジェクトは，それを保持するユーザが CaStor からログアウトした後に CaStor によってすべて削除される．これにより，CaStor を介さないでプロキシにアクセスすること

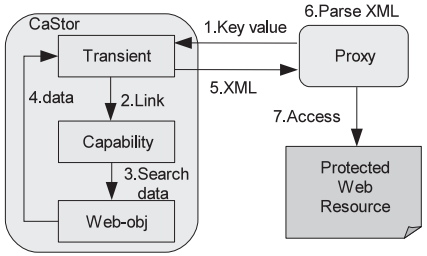

図 5　一時的オブジェクトを介したデータの取得
Fig. 5　Getting data through a transient object.

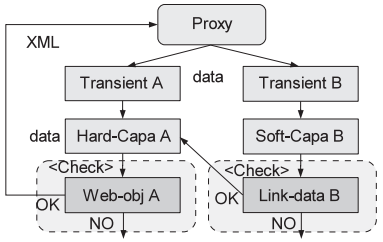

図 6　有効期限と使用回数制限の検証
Fig. 6　Validation of expiration date and limited amount of uses.

ができないため，上述した問題を解決することができる．また，プロキシへの URL が他のユーザに漏洩した場合，その URL が他のユーザに使用されセッションハイジャックされる可能性がある．これを防ぐために，CaStor ではユーザが CaStor にログインしたときにクッキーをブラウザに渡し，プロキシが CaStor から情報を取り出すときには URL の乱数だけではなくクッキーの値も使用する．

一時的オブジェクトとクッキーを用いた保護された Web 資源へのアクセス方法を**図 5** に示す．CaStor・プロキシ間の通信は，REST（REpresentational State Transfer）を用いて実装した．ユーザが，ある保護された Web 資源にアクセスを行う場合，その Web 資源に対応するケーパビリティを選択する．このとき，CaStor は，一時的オブジェクトを作成し，それにアクセスするための乱数を含むプロキシへの URL をブラウザに返す．ブラウザは，その URL を用いてプロキシにアクセスすると同時にプロキシにクッキーを送信する．プロキシは，HTTP GET メソッドを用いて URL に付加された乱数とクッキー内の値（乱数）を CaStor に送信する．CaStor は，受信したキー値に一致する一時的オブジェクトを検索し，一時的オブジェクトの保持するケーパビリティの識別子を用いて対応する Web オブジェクトの識別子を取り出す．その後，CaStor は，その識別子を用いて Web オブジェクトからアクセスするために必要なデータを XML 形式でプロキシに返す．プロキシは，受け取った XML から必要な情報を取り出し保護された Web 資源にアクセスを行う．

3.4　有効期限や使用回数制限の実現

図 6 に有効期限と使用回数制限の検証方法について示す．まず，有効期限となる時刻や使用回数の上限を Web_object テーブル，あるいは，Link_data テーブルに保存する．ユーザが，ケーパビリティを用いて外部の保護された Web 資源にアクセスを行う場合，3.3 節で述べたように対応する一時的オブジェクトが作成される．図 6 では，一時的オブジェクト A とハード・ケーパビリティA，そして，一時的オブジェクト B とソフト・ケーパビリティB が対応している．ハード・ケーパビリティA とソフト・ケーパビリティB は，どちらも Web オブジェクト A を参照している．

ハード・ケーパビリティA の場合，Web オブジェクトからデータを取り出す際に，有効期限と使用回数制限が検証される．検証が通った場合，データは CaStor からプロキシに送信される．このとき，使用回数の上限が 1 減らされる．検証が通らなかった場合，エラーが返される．

ソフト・ケーパビリティB の場合，Web オブジェクト A の前にリンク・データ B で有効期限と使用回数制限が検証される．検証が通った後，リンク・データ B が保持するハード・ケーパビリティA の識別子を用いて．ハード・ケーパビリティA を参照する．このとき，リンク・データ B の使用回数の上限が 1 減らされる．検証が通らなかった場合，エラーが返される．その後，上述したハード・ケーパビリティA の場合と同様の動作で検証が行われる．Web オブジェクト A でも検証が通った場合，データは CaStor からプロキシに送信される．

3.5　インボックスを用いたケーパビリティの配布の実装

ケーパビリティを配布する場合，ユーザは，自分の保持するケーパビリティの中から配布したいケーパビリティを選択する．そうすると，ブラウザは CaStor の持つ配布機能の画面に移動する．そこで，ユーザは，“To” に送信先ユーザのユーザ ID を入力し “送信” ボタンを押す．このとき，何のためのケーパビリティであるか等のメッセージを “Body” に入力することができる．CaStor は，指定されたケーパビリティの capability_ID を指定されたユーザの user_ID と関連付けられた Inbox テーブルに保存する．インボックスからディレクトリにケーパビリティを移動させる場合，受け取ったユーザは，インボックス内から保存したいケーパビリティを選択し保存先のディレクトリを指定する．その後，CaStor は，イ

ンボックス内の指定されたケーパビリティの capability_ID を用いて取り出したデータと指定したディレクトリの directory_ID を Capability テーブルに保存する．

3.6　フォーム認証用プラグイン

現在，CaStor は，Basic 認証に関してはプロキシの機能[13] をそのまま利用して使用可能である．これに対して，アクセス対象の保護された Web 資源がフォーム認証を用いている場合に対応するためにプラグインを用意する．このプラグインは，フォーム認証における Web 資源ごとにフォームのデータの違いを吸収する．たとえば，ある Web 資源では，ユーザ ID として “email” を使用している．また，ある Web 資源では，ユーザ ID として “username” を使用している．このようなフォームのデータの違いに対応するために，各 Web 資源のフォームに対応したプラグインが必要になる．

プラグインは，データの違いに対応するだけではなく，Web 資源からのクッキーをユーザの代わりに受け取ることも行う．プラグインが受け取ったクッキーはプロキシを通して CaStor に保存され，プロキシがその Web 資源にアクセスする際に Web 資源に送信される．

4.　実　　験

この章では，CaStor を用いて保護された Web 資源にアクセスする場合の処理時間を計測した実験について述べる．

CaStor では，保護された Web 資源にアクセスする場合データベースやプロキシを利用するため，直接アクセスを行う場合よりも通信に時間がかかる．そのため，CaStor を用いて保護された Web 資源にアクセスを行った場合の処理時間を計測し，実用に問題がないことを示す．

4.1　実 験 方 法

本実験では，CaStor にログイン後，ケーパビリティを用いて初めて保護された Web 資源にアクセスを行う場合を初回アクセス時とする．プロキシは，初回アクセス時に 3.3 節で述べたように REST を用いて CaStor から保護された Web 資源にアクセスするためのデータを受け取り，キャッシュ機能を用いてユーザが CaStor にログインしている間は保持し続ける．2 回目以降のアクセスは，CaStor からログアウトをしないで保護された Web 資源にアクセスを行うこととしている．また，キャッシュされたデータは，ユーザが CaStor をログアウトする際にプロキシによって消去される．初回のアクセスでは，HTML ファイルの取得を想定している．2 回目以降のアクセスでは，その HTML ファイルに含まれる画像の表示や他の HTML ファイルへの移動を想定している．

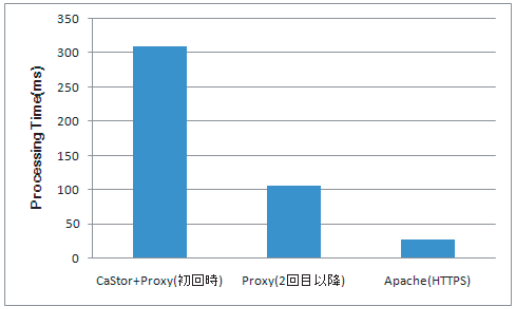

図 7　実験結果
Fig. 7　Experimental results.

本実験では，以下の 3 つの処理時間を計測する．

- 初回時：CaStor とプロキシを用いて保護された Web 資源を取得する（**図 7** の CaStor + Proxy）．
- 2 回目以降：プロキシのみで保護された Web 資源を取得する（図 7 の Proxy）．
- 参考値：Apache サーバと HTTPS を用いて通信を行い HTML ファイルを取得する（図 7 の Apache（HTTPS））．

処理時間は，Apache Bench を用いて 1 コネクションに 1 リクエストを送信して取得した．プロキシは Java で実装されているため，本実験では，パフォーマンスを向上させるために実行時に HotSpot Server VM を用いる．また，HotSpot Server VM の JIT（Just In Time）コンパイルやメモリの影響を減らすために，“-XX:CompileThreshold=100”，“-Xms512m”，“-Xmx512m”，や “-XX:NewSize=256m” オプションを用いて，実際に処理時間を計測する前にプロキシを 200 回実行し，よく使用するコードの JIT によるコンパイルを行った．

本実験では，以下の 3 台のマシンを使用する．

- クライアント用．
- CaStor とプロキシ用．
- 保護された Web 資源用．

実際の Web サイトにアクセスを行う場合，インターネットの通信遅延が大きなものになり実際に保護された Web 資源へのアクセスにかかった処理時間が分らない可能性がある．そのため，CaStor と同一 LAN 内にテスト用としてベーシック認証により保護された Web 資源，要求を投げるクライアントを設置した．保護された Web 資源のサイズは，32 KB と

表 1　実験環境
Table 1　The environment of experiments.

CPU	Intel Xeon 3.6 GHz
Memory	1.0 GB
OS	Debian GNU/Linux (Kernel 2.6.18)
LAN	1000BASE-T
Client	Apache Bench 2.0.40
Language Systems	Ruby1.8, Rails2.0.2, Java1.5
Web server	Apache2.2.3

した．実験に使用したマシン環境を**表 1** に示す．

4.2 実験結果

実験結果を図 7 に示す．実験結果は，処理時間を 10 回計測し平均をとったものである．まず，同じ環境で Apache サーバと HTTPS を用いて通信を行いプレーン HTML を取得するのにかかった処理時間を計測した．その処理時間は，約 27 ms であった（図 7 の Apache (HTTPS)）．

初回アクセス時の処理時間（図 7 の CaStor + Proxy）は，約 309 ms だった．これは，Web 資源に初めてアクセスする場合，利用者認証を通るため，CaStor の保持するデータベースにアクセスしデータを取り出す必要がある．そのとき，CaStor とプロキシ間で通信を行う際データベースアクセスやブラウザ・CaStor 間とブラウザ・プロキシ間の両方で HTTPS を用いているため，処理時間が大きくなったと考えられる．しかし，インターネットの通信遅延と比較すると，問題ない処理時間だといえる．

2 回目以降のアクセス（図 7 の Proxy）の処理時間は，約 106 ms だった．2 回目以降にアクセスするときは，プロキシがアクセスに必要なデータをキャッシュしているため，プロキシのみを介してクライアントと Web 資源間で通信が行われる．Apache サーバとの HTTPS 通信よりも処理時間が増加したのは，プロキシが Web 資源からの応答を中継するためだと考えられる．こちらも，インターネットの通信遅延と比較すると，問題ない処理時間だといえる．

これらの結果から，CaStor を用いて保護された Web 資源にアクセスする処理時間は，インターネットの通信遅延と比べて問題ない程度の大きさであり実用性があるといえる．

5. 関連研究

ケーパビリティに基づくアクセス制御は初期のマルチプロセッサや分散オペレーティング・システムの研究で使用された．ケーパビリティに基づくアクセス制御を用いることで，プロセス間で権限の委譲を行うことが可能になる．Hydra では，ケーパビリティはオブジェクトへの参照として使用された[24]．Mach では，ケーパビリティは通信ポートのアクセスを制御するために使用された[1]．Amoeba は，ユーザ・プロセスが一般的なプロセス間通信を用いてケーパビリティを受け渡すことを許可している[17]．近年，携帯電話のためのオペレーティング・システムとして開発された OS である Symbian OS は，資源の保護をするためにケーパビリティを使用する[21]．また，複数のマシン上にあるデータベースへのアクセスにケーパビリティを用いる HomeViews という研究がある[5]．HomeViews では，ケーパビリティをユーザ間で交換することでデータを共有することを可能にしている．CaStor は，これらのシステムと比較してインターネット上にある Web 資源を対象とする点が異なる．

ケーパビリティを用いたシステムにおいて，ケーパビリティの偽造を防ぐことは重要である．ケーパビリティの偽造を防ぐために，Hydra や Mach ではケーパビリティを OS カーネル内に格納し，Amoeba では一方向関数を用いて暗号化している．Symbian OS では，ケーパビリティは実行可能ファイルに組み込まれ変更することはできない．HomeViews では，ケーパビリティに乱数を含ませることで類推されないようにしている[3),14)]．CaStor では，ケーパビリティを 1 つのサーバで集中管理するため，Mach や Hydra の方法に近い．

権限の委譲はケーパビリティに基づくアクセス制御において実現されていたが，近年，それ以外のアクセス制御の研究においても権限の委譲を許すことが試みられている．社会的な関係をユーザ間に持ち込むことで，権限の委譲を可能にするアクセス制御方式の研究がある[11]．これは，社会的関係に応じた権限フィルタとすでに権限を保持しているユーザの権限の積集合をとることで，一時的な権限を作成しそれを他のユーザに渡すことで権限の委譲を可能している．ただし，活動の場に社会的関係のあるユーザがいない場合，たとえそのユーザのことを知っているユーザがいたとしても権限を委譲することはできない．本研究で用いているケーパビリティに基づくアクセス制御では，社会的関係による制約はなく，ケーパビリティを渡すことで権限を委譲することが可能である．

システムがすでに信頼しているユーザが，他のユーザを信頼するとシステムがそのユーザを間接的に信頼するという信頼の輪モデルという研究がある[16]．これは，事前にシステムに登録（信頼）されているユーザが，新しいユーザをシステムに登録することができるというものである．新しいユーザのシステムに対する権限は，自分を信頼してくれたユーザの権限以下になる．この研究とケーパビリティに基づくアクセス制御は，あるユーザが信頼しているユーザを間接的に信頼するという点で類似している．しかし，この研究は ACL を用い

るためにアクセスの主体を識別するという点において，ケーパビリティに基づくアクセス制御とは異なる．

また，プロキシを用いてアクセス制御の設定を行う権限をユーザ間で委譲可能にしたシステムがある[19]．このシステムでは，管理者でなくてもこの権限を持っているユーザであれば誰でも，アクセス権限を設定することが可能である．このシステムと CaStor は，プロキシを用いることで Web 資源に対するアクセス制御を実現している点で類似している．しかし，このシステムのプロキシは ACL を用いているのに対して，CaStor のプロキシはケーパビリティを用いている点で異なる．

そのほかにも，自分の持つロールやそのロールに含まれるパーミッションを他のユーザやロールに委譲可能なロールに基づくアクセス制御モデル（PBDM: Permission-based Delegation Model）が提案されている[23]．このモデルを用いることで，一時的なロールを作成しそのロールにロール，あるいは，パーミッションを付加し他のユーザに委譲することで，アクセス権限を容易に委譲することが可能になる．PBDM はロールに基づくアクセス制御モデルを用いているためアクセスの主体を識別する必要がある．これに対して，ケーパビリティに基づくアクセス制御では，ケーパビリティ自体に自己認証機能があるためアクセスの主体を識別する必要はない．

Keychain Access は，ログインしたユーザのパスワードやフォームを効率良く管理するためのアプリケーションである[9]．Keychain Access と連携した Web ブラウザでは，フォームデータやパスワードが自動的に入力される．1Password は，Keychain Access のパスワード入力機能を強化し，さらにパスワードの自動生成機能を付加したものである[2]．Keychain Access が 1 種類の Web ブラウザでしか利用できないのに対して，1Password は複数の種類の Web ブラウザで利用可能である．また，my1Password という Web サービスを用いることでネットワークを介して登録したパスワードを取り出すことができる．ただし，Keychain Access を用いているため，MacOS X 以外の OS で使用することができない．MacOS X の Keychain Access と比較して，CaStor では，ネットワークにつながっているマシンならばどこからでもケーパビリティを取り出すことが可能な点で異なる．また，Keychain Access や 1Password と my1Password は，個人のパスワードを管理することを目的としており，アクセス権限を他のユーザに渡すことは考慮されていない．それに対して，CaStor は，パスワードの管理ではなくアクセス権限を他のユーザに委譲することで作業分担の支援を行うことに重点を置いている．

Kerberos は，シングルサインオンを実現するためのネットワーク認証システムである[20]．Kerberos ではユーザ認証が行われると，TGT（Ticket Granting Ticket）と呼ばれる値が得られる．一般のサービス，たとえば，遠隔ログインやメールボックスへのアクセスを受けるときには，TGT から生成したチケットを示すことで個別の重複したユーザ認証を避けることができる．Liberty Alliance は，World Wide Web においてシングルサインオンを実現するための仕組みである[8]．Liberty Alliance では，複数の連携した Web サイト間でユーザ認証の情報を交換することができる．OpenID は，シングルサインオンを実現するためのプロトコルである[12]．OpenID に対応した Web サイトであれば，共通した 1 つの ID で認証を受けることができる．このような，シングルサインオンの仕組みはケーパビリティの仕組みに類似しているが，シングルサインオンの場合，ユーザ認証機能を省略するための機能しかない．これらのシングルサインオンのプロトコルと比較して，CaStor は，アクセス権限の作成や他のユーザへの配布といった機能を持つ．

Google API[6]，Flickr API[4]，そして，OAuth protocol[10] は，保護された Web 資源への認証を支援する API とプロトコルである．Google API，Flichr API，そして，OAuth プロトコルは，トークンを使用して外部の保護された Web 資源に外部の Web 資源がアクセスすることを許可する．ある外部の Web 資源が，これらの API やプロトコルに対応した Web サイト内にある保護された Web 資源にアクセスしようとした場合，Web サイトはその資源の所有者であるユーザに認証情報の入力を求める．Web サイトは，その認証情報をもとに作成したトークンを外部の Web 資源に渡す．このとき，一時的に使用可能なトークンを作成することが可能である．その後，外部の Web 資源は，そのトークンを用いて保護された Web 資源にアクセスを行うことが可能になる．これらのシステムでは，あるユーザとある Web 資源間でのアクセス権限の受け渡しが可能だが，ユーザ間でのアクセス権限の受け渡しはできない．

6. ま と め

本論文では，既存の Web 資源に対するケーパビリティの管理・配布を行う Web サーバである CaStor について述べた．CaStor は，保護された Web 資源にアクセスするために必要な情報（ユーザ ID，パスワード，そして，Web 資源の URL）からケーパビリティを作成し保存し，他のユーザに安全に配布する機能を持つ．また，ケーパビリティには，アクセス可能な URL のパターン，有効期限，そして，使用回数制限を付加することができる．そのため，CaStor を用いることで，既存の保護された Web 資源のプログラムを変更することなく，他のユーザにアクセス権限の一部を渡すことが可能になるためユーザ間での作業分

担が可能になる．CaStor では，機密情報を他のユーザに渡すことなく保護された Web 資源にアクセスを行うことを可能にするためにプロキシを用いた．これにより，プロキシの持つ機能を利用してアクセス可能な Web 資源を制限することも可能である．我々は，CaStor をインターネット経由でアクセス可能にするため，Ruby on Rails を用いて Web サーバとして実装した．ブラウザ・CaStor 間とブラウザ・プロキシ間で盗聴を防ぐため，通信には HTTPS を用いている．また，CaStor を用いて保護された Web 資源にアクセスするまでの処理時間を計測し，実用に問題がないことを示した．

CaStor は，Web サーバでありインターネット上で複数稼働させられることが考えられる．そのため，CaStor どうしが連携し，ある CaStor から他の CaStor にケーパビリティを配布できるようにすることが今後の課題である．また，Web 資源ごとのフォームのデータを吸収するためにプラグインを用意するのではなく，自動的に解釈しアクセス可能にすることも今後の課題である．

参 考 文 献

1) Accetta, M., Baron, R., Bolosky, W., Golub, D., Rashid, R., Tevanian, A. and Young, M.: Mach: A New Kernel Foundation For UNIX Development, *Proc. USENIX Summer Conference*, pp.93–112 (1986).
2) Agile Web Solutions: 1Password User Guide, Agile Web Solutions (online). http://agilewebsolutions.com/products/1Password/user_guide (accessed 2008-12-03)
3) Anderson, M., Pose, R.D. and Wallace, C.S.: A Password-Capability System, *The Computer Journal*, Vol.29, No.1, pp.1–8 (1986).
4) Flickr: Flickr Authentication API Specification — Revision 1.1.9, Flickr (online). http://flickr.com/services/api/auth.spec.html (accessed 2008-12-03)
5) Geambasu, R., Balazinska, M., Gribble, S.D. and Levy, H.M.: HomeViews: Peer-to-Peer Middleware for Personal Data Sharing Applications, *Proc. 2007 ACM SIGMOD International Conference on Management of Data*, pp.235–246 (2007).
6) Google: Google Account Authentication API Authentication for Web Apps, google (online). http://code.google.com/apis/accounts/Authentication.html (accessed 2008-12-03)
7) Levy, H.M.: *Capability-Based Computer Systems*, Digital Press (1984).
8) Liberty Allience Project: INTRODUCTION TO THE LIBERTY ALLIANCE IDENTITY ARCHITECTURE Revision 1.0, Liberty Allience Project (online). http://xml.coverpages.org/LibertyAllianceArchitecture200303.pdf (accessed 2008-12-031)
9) Apple ADC: Introduction to Keychain Services Programming Guide, ADC (online). http://www.monen.nl/DevDoc/documentation/Security/Conceptual/keychainServConcepts/01introduction/chapter_1_section_1.html (accessed 2008-12-03)
10) OAuth: OAuth Core 1.0, OAuth (online). http://oauth.net/core/1.0/ (accessed 2008-12-03)
11) 小川悟史，ラデスクジョルジュ・シェルパン，魏　文鵬，北形　元，武田敦志，白鳥則郎：現実空間での社会的振舞を活用した柔軟かつ安全なアクセス制御方式，電子通信学会情報ネットワーク研究会，Vol.107, No.222, pp.137–142 (2007).
12) OpenID: OpenID Authentication 2.0, OpenID (online). http://openid.net/specs/openid-authentication-2_0.html (accessed 2008-12-03)
13) 松井慧悟，佐藤　聡，新城　靖，板野肯三，馬渕充啓，加藤和彦：Web ページに対するケーパビリティを用いたアクセス制御のプロキシによる実現，情報処理学会システムソフトウェアとオペレーティングシステム研究会，2007-OS-105, pp.95–10 (2007).
14) Mabuchi, M., Shinjo, Y., Sato, A. and Kato, K.: An Access Control Model for Web-Services that Supports Delegation and Creation of Authority, *Proc. 7th International Conference on Networking* (*ICN08*), pp.213–222 (2008).
15) 馬渕充啓，池嶋　俊，川崎仁嗣，吉野純平，松井慧悟，新城　靖，佐藤　聡，加藤和彦：既存の Web 資源に対するケーパビリティの管理・配布を行うサーバの実現，情報処理学会シスイテムソフトウェアとオペレーティングシステム研究会，2008-OS-107, pp.41–48 (2008).
16) 正岡　元，菊池　豊：信頼の輪モデルに基づいたシステム利用権限の委譲による個人認証手法，情報処理学会分散システム/運用技術研究会，2003-DSM-29, pp.51–56 (2003).
17) Mullender, S.J., Rossum, van G., Tenenbaum, A.S., Renesse, van R. and Staveren, van H.: Amoeba: A Distributed Operating System for the 1990s, *IEEE Computer*, Vol.23, No.5, pp.44–53 (1990).
18) Ruby on Rails: Rails Framework Documentation, Ruby on Rails (online). http://api.rubyonrails.org/ (accessed 2008-12-03)
19) 関口聖美，黒羽秀一，齋藤孝道：ユーザによる設定を可能とする Proxy 型ネットワークアクセス制御方式の提案，情報処理学会コンピュータセキュリティ研究会，2007-CSEC-37, pp.39–44 (2007).
20) Steiner, J.G., Neuman, B.C. and Schiller, J.I.: Kerberos: An Authentication Service for Open Network Systems, *Proc. Winter 1988 Usenix Conference*, pp.191–201 (1988).
21) Stichbury, J. and Jacobs, M.: *The Accredited Symbian Developer Primer: Fundamentals of Symbian OS*, John Wiley & Sons Inc. (2006).
22) Goland, Y., Whiteheadm E., Faizi, A., Carter, S. and Jensen, D.: HTTP Extensions for Distributed Authoring – WebDAV, Network Working Group Request for

Comments: RFC 2518 (1999).
23) Xinwen, Z., Sejong, O. and Ravi, S.: PBDM: a flexible delegation model in RBAC, *Proc. 8th ACM Symposium on Access Control Models and Technologies*, pp.149–157 (2003).
24) Wulf, W., Cohen, E., Corwin, W., Jones, A., Levin, R., Pierson, C. and Pollack, F.: HYDRA: The Kernel of a Multiprocessor Operating System, *Comm. ACM*, Vol.17, No.6, pp.337–345 (1974).

(平成 20 年 12 月 4 日受付)
(平成 21 年 5 月 13 日採録)

馬渕　充啓（学生会員）

昭和 57 年生．平成 17 年筑波大学第三学群情報学類卒業．平成 19 年筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士前期課程修了．現在，同博士後期課程 3 年．修士（工学）．オペレーティングシステム，分散システム，ネットワークセキュリティ，アクセス制御に興味を持つ．

池嶋　　俊

昭和 58 年生．平成 18 年筑波大学第三学群情報学類卒業．平成 20 年筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士前期課程修了．分散システム，ネットワーク，P2P システムに興味を持つ．

川崎　仁嗣

昭和 58 年生．平成 18 年筑波大学第三学群情報学類卒業．平成 20 年筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士前期課程修了．現在，（株）NTT ドコモ．ユーザインタフェースの研究に従事．システムソフトウェア，特にコンピュータセキュリティに興味を持つ．

吉野　純平

昭和 58 年生．平成 18 年筑波大学第三学群情報学類卒業．平成 20 年筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士前期課程修了．広域環境でのデータ転送技術の研究に従事．

松井　慧悟（正会員）

昭和 60 年生．平成 21 年筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士前期課程修了．現在，（株）日立製作所．大学では，アクセス制御の研究に従事．

新城　　靖（正会員）

昭和 40 年生．平成 5 年筑波大学大学院博士課程工学研究科電子・情報工学専攻修了．同年琉球大学工学部情報工学科助手．平成 7 年筑波大学電子・情報工学系講師，平成 15 年同助教授．平成 19 年准助教授．オペレーティング・システム，分散システム，仮想システム，並行システム，情報セキュリティの研究に従事．博士（工学）．日本ソフトウェア科学会，ACM，IEEE CS 各会員．

佐藤　　聡（正会員）

昭和 42 年生．平成 3 年筑波大学第三学群情報学類卒業．平成 8 年筑波大学大学院工学研究科単位取得退学．同年広島市立大学情報科学部助手．平成 13 年筑波大学システム情報工学研究科講師．現在，同大学学術情報メディアセンター勤務．博士（工学）．キャンパスネットワークの企画管理運用，ネットワークデータベース，言語処理等の研究に従事．電子情報通信学会，ACM-SIGMOD-JAPN 各会員．

上川　大介

昭和 59 年生．平成 15 年神奈川大学理学部情報科学科入学．平成 20 年神奈川大学理学部情報科学科卒業．現在，筑波大学大学院システム情報工学研究科コンピュータサイイエンス専攻所属．

加藤　和彦（正会員）

昭和 37 年生．昭和 60 年筑波大学第三学群情報学類卒業．平成元年東京大学大学院理学系研究科情報科学専攻中退．平成 4 年博士（理学）(東京大学大学院理学系研究科)．平成元年東京大学理学部情報科学科助手，平成 5 年筑波大学電子・情報工学系講師，平成 8 年同助教授，平成 16 年筑波大学大学院システム情報工学研究科教授，現在に至る．オペレーティングシステム，分散システム，仮想計算環境，セキュリティに興味を持つ．平成 2 年情報処理学会学術奨励賞，平成 4 年同研究賞，平成 17 年同論文賞，平成 16 年日本ソフトウェア科学会論文賞各受賞．