CaStor: 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

馬渕充啓　　池嶋俊　　川崎仁嗣
吉野純平　　松井慧悟

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・配布を行うサーバ (CaStor) について述べる．CaStor は，大きく 3 つの機能を持つ．第 1 に，ユーザ名とパスワードにより認証を行うような既存の Web 資源にアクセスを行うためのユーザ名とパスワードを，利用者に代わり一元的に管理する機能を持つ．第 2 に，そのような既存の Web 資源に対して，それにアクセスできるようなケーパビリティを発行する機能を持つ．第 3 に，発行したケーパビリティを他の利用者に安全に配布する機能を持つ．

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

Mitsuhiro Mabuchi, Shun Ikejima, Satoshi Kawasaki,
Junpei Yoshino and Keigo Matsui

In this paper, we describe about CaStor, a server for management and distribution of capabilities to access existing Web resources. CaStor has three main functions. The first function is to manage user names and passwords on behalf of users to access existing Web resources that requires user authentication by using user names and passwords. The second function is to create capabilities to access these such existing Web resources. The third function is to distribute capabilities to other users in a secure way.

1. はじめに

Google，Yahoo，そして，Amazon 等に代表される Web サイトは多くの Web 資源を提供している．たとえば，メールボックス，カレンダー，あるいは，買い物かご等があげられる．これらの Web 資源のうち個人に属するものは，多くの場合，ユーザ ID とパスワードを用いて利用者認証により保護されている．その結果，ユーザは大量の異なるユーザ ID やパスワードを管理しなければならなくなってきている．近年，ユーザの保持する Web 資源にアクセスするためのユーザ ID とパスワードといった認証情報の量はユーザの管理能力を超えており，その管理方法は大抵煩雑になることが多い．

これらの問題を解決するため，パスワード管理アプリケーションやシングルサインオンのしくみが開発された．パスワード管理アプリケーションを使用することで，パスワード管理や入力といった手間を省くことができる．また，シングルサインオンに対応した Web 資源の場合，1 度のユーザ ID とパスワードの入力により複数の Web サイトを利用することができる．しかし，これらの方法は次の 3 つの問題点を持つ．

- パスワード管理アプリケーションでは Web 資源への個人のユーザ ID とパスワードをローカルな場所に管理するため，他のマシンから利用することができない．
- これらは，自分のアクセスできる資源のアクセス権限から別のアクセス権限を作成することができない．
- そのアクセス権限を他のユーザに渡す機能がない．

このような問題を解決するため，われわれは既存の Web 資源に対してケーパビリティに基づくアクセス制御を導入している[4)10)]．本論文では，ケーパビリティの管理と他ユーザへの配布を行うことができるサーバ (CaStor) について述べる．CaStor は，大きく次の 3 つの機能を持つ．第 1 に，ユーザ名とパスワードにより認証を行う既存の Web 資源にアクセスを行うためのユーザ ID とパスワードを，利用者に代わり一元

的に管理する機能を持つ．第 2 に，そのような既存の Web 資源に対して，それらにアクセスできるケーパビリティを発行する機能を持つ．第 3 に，発行したケーパビリティを他の利用者に安全に配布する機能を持つ．これらの機能を利用することで，ある利用者がある Web 資源に対するケーパビリティを作成し他の利用者に渡すことで，その利用者が元の Web 資源にユーザ登録されていなくてもアクセスすることができるようにする．それらの機能を提供する CaStor は，ケーパビリティを管理するマネージャ・プログラムと Web 資源へのアクセス時に使用するプロキシ[9] から構成される．

本論文は，以下のように構成される．2 章では，既存のパスワード管理の問題点とケーパビリティを用いる利点について述べる．3 章では CaStor の設計について述べ，4 章では実装について述べる．5 章では，関連研究について述べる．そして，6 章ではまとめを行う．

2. 既存のパスワード管理の問題点とケーパビリティを用いる利点

従来，ユーザ ID とパスワードを管理する方法として，ユーザが自ら覚えたり（暗記，あるいは，メモ等）ブラウザに記憶させる等の管理方法が取られている．自ら覚える場合，パスワードの消失やパスワードを頻繁に変更する度に手間がかかる．また，ブラウザに記憶させる場合，自分ではパスワードを忘れることが多いため別マシンでパスワードを入力できない等の問題が起こる可能性がある．このように，大量のパスワードを保持しているユーザのパスワード管理は煩雑になりやすい．ユーザ ID やパスワードの取扱いに関して，ユーザは以下のような要求を持つ．

- ユーザ ID やパスワードの管理を簡単に行いたい: たとえば，パスワードの記憶や入力を省略したり，複数の場所で使用したいので個々の Web ブラウザのパスワード管理機能を使用したくないという要求がある．
- ユーザ登録をしないで Web 資源にアクセスしたい: たとえば，ユーザ登録の手間を省略したいという要求や組織外の未登録ユーザにも，保護された Web 資源にアクセスさせたいという要求がある．
- 自分が持つアクセス権限を元に，その一部のアクセス権限を他のユーザに渡したい: たとえば，私設秘書に自分のアカウントで会議室予約やスケジュール管理を依頼したいが，自分の全てのアクセス権限を渡したくないという要求がある．

それらの問題解決のため，最近では，MacOS X の Keychain Access[6] 等のパスワード管理アプリケーションが使用されている．しかし，これらのアプリケーションは個人のユーザ ID とパスワードをローカルな場所に管理するため，他のマシンから利用することができない．そのため外出先のマシンから Web 資源にアクセスしようとした場合，パスワードを覚えていないためアクセスできないということが起こる．

アプリケーション以外にも，Liberty Alliance[5]，OpenID[8]，あるいは，Kerberos[15] 等の 1 つのユーザ ID とパスワードで複数の Web 資源へのアクセスを可能にするシングルサインオン・プロトコルがある．これらのプロトコルに対応している複数の Web 資源では，1 度あるサイトでユーザ ID とパスワードにより認証を受けた後は認証なしに他のサイトを利用することができるため非常に有用である．しかし，現状ではまだ普及していないので対応していない Web 資源が多い．

ユーザ ID とパスワードの管理が煩雑になりやすい以外にも，既存の Web 資源へのアクセス制御方式には協調作業を阻害する問題がある．大抵の Web 資源では，ユーザ登録を行わなければ使用できないため，登録不可能な外部ユーザはアクセスできない．そのため，従来，外部ユーザに作業を頼みたい場合，自分のユーザ ID とパスワードを渡すという運用を行っていることがある．この場合，その外部ユーザはそのアカウントで可能な権限を全て保持することになる．1 つの作業を頼むためだけに全ての権限を与えることは避けたいことである．

これらの問題点を解決するために，CaStor は以下の 3 つの機能を提供する．第 1 に，CaStor は，ユーザ ID とパスワードをネットワークを介してアクセス可能な場所で一元管理する機能を提供する．これにより，外出先からでもユーザ ID とパスワードを取り出して目的の Web 資源にアクセスすることが可能になる．また，特別なプロトコルは用いていないため Web 資源側が CaStor に対応する必要はない．第 2 に，CaStor では，ユーザ ID やパスワードを元にケーパビリティを作成する機能を提供する．たとえば，有効期限や使用回数制限の付加やある Web サイトのある Web ページにのみアクセス可能にしたケーパビリティを作成することができる．CaStor では，ユーザ間で Web 資源へのアクセス権のやり取りを行うことを可能にするためアクセス制御にケーパビリティ[4][10] の概念を用いる．ケーパビリティとは，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のリストのこ

とである．第 3 に，CaStor は，作成したケーパビリティを他のユーザに安全に渡す機能を提供する．これにより，ある作業をあるユーザに依頼したい場合，制限を付加したケーパビリティを渡すことで全ての権限を渡す必要がなくなる．また，ある Web 資源の登録ユーザが新たなケーパビリティを作成し未登録ユーザに渡すことで未登録でも Web 資源にアクセスし作業を行うことができる．この時，登録ユーザは，ユーザ ID とパスワードを未登録ユーザに渡すことはない．

3. CaStor の設計

2 章で述べた機能を達成するためインターネットから Web ブラウザでアクセス可能なサーバを設置する．1 つのサーバにおいてケーパビリティを集中管理するため，ユーザ間でケーパビリティのやり取りを容易に行うことができる．また，サーバをインターネットでアクセスできる場所で稼働させることで外出先からでも利用可能にすることができる．

CaStor を利用する場合，ユーザ (利用者) は自分のケーパビリティが保存されているスペースにアクセスするため認証を受ける (ログインする) 必要がある．CaStor では，ユーザはケーパビリティを保存する複数のスペースを持つことができる．たとえば，仕事用，あるいは趣味用等の用途別に複数のスペースを使い分けることができる．CaStor では，それらのスペースを**ケーパビリティ・セット** と呼びケーパビリティはそれぞれのセットに属して保存される．ログインすることにより，CaStor は，セットの識別子を受け取りそれをもとにそのセットに属したケーパビリティをブラウズする．

CaStor では，以下のオブジェクトを扱う．

- セット: ケーパビリティの集合のことで，ユーザは複数使用することが可能である．
- ケーパビリティ: Web 資源にアクセスするために必要な URL，ユーザ ID とパスワード，および，有効期限や使用回数制限等を含む．
- 間接ケーパビリティ: ケーパビリティを間接的に使用するためのケーパビリティのことである．間接ケーパビリティ自体にも有効期限や使用回数制限の設定が可能である．
- 受信箱: 受け取ったケーパビリティを一時的に保存する．1 つのセットにつき 1 つの受信箱が存在する．

図 1 に CaStor のモデルを示す．まず，ユーザはネットワークを介して CaStor にアクセスする．次に，自分のセット (点線の枠) 内からアクセスしたい Web 資源へのケーパビリティを選択しプロキシ[9] に渡す．プロキシは，受け取ったケーパビリティを元に Web 資源へのオリジナルな URL を復元する．次に，プロキシは，その URL を元に Web 資源へ要求を送り返ってきた応答 (ステータスコード，ヘッダ，メッセージボディ) をそのままユーザに返す．CaStor では，このプロキシを用いて既存の保護された Web 資源にアクセスする．保護された Web 資源とは，利用者認証によりアクセス制限を行っている資源のことである．また，このプロキシを用いることで Web サイト内の一部のページにのみアクセス可能なケーパビリティを作成することができる．たとえば，SNS サイト内にある特定の日記のみ閲覧可能なケーパビリティ等である．

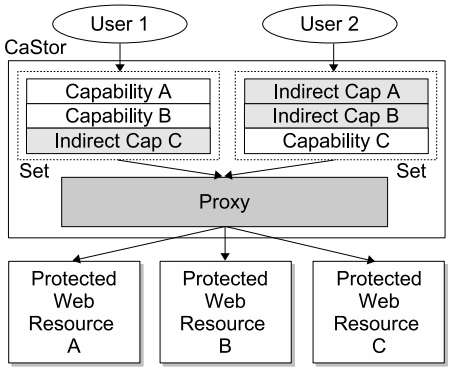

図 1 CaStor のモデル

3.1 ケーパビリティのブラウズと使用方法

図 2 に Web ブラウザを用いたケーパビリティのブラウズ画面を示す．ここには，ユーザがログインしたケーパビリティ・セットに属しているケーパビリティが表示される．ユーザは，この中からアクセスしたい Web 資源を選択する．CaStor は，選択された Web 資源にアクセスするために必要なケーパビリティを取り出し使用する．

3.2 ケーパビリティの管理

CaStor では，ケーパビリティを次の 7 つの手続きで管理する．

- **作成**: ユーザから保護された Web 資源の URL，ユーザ ID，パスワード，および，有効期限等を受け取りケーパビリティを作成・暗号化し，指定されたセットに保存する．また，間接ケーパビリティを作成することもできる．
- **編集**: ケーパビリティの内容を編集して保存する．編集内容として，ケーパビリティ名，有効期限，そして，使用回数制限等がある．
- **削除**: ケーパビリティを削除する．
- **コピー**: ケーパビリティをコピーする．

図 2 ケーパビリティのブラウズ画面

- **送信**: 他のセットの受信箱にケーパビリティを送信する．送信するものはケーパビリティと間接ケーパビリティの 2 種類ある．
- **受信**: 受信箱の中から必要なケーパビリティを選択し，受信箱を保持するセットに受信し保存する．
- **表示**： ケーパビリティを使用しやすい形で表示する．

3.3 ケーパビリティの暗号化

ケーパビリティは機密情報であるため，データベースに保存する際に暗号化する必要がある．暗号化する手法として共通鍵暗号方式を用いる．

CaStor では，共通鍵をケーパビリティ毎に作成し，その鍵を用いて機密情報を暗号化する．また，作成された共通鍵はセットに関連付けて保存する．ケーパビリティを暗号化した共通鍵は別の共通鍵で暗号化し，その共通鍵はセットにアクセスするためのパスワードで暗号化する．これにより，パスワードを変更した場合，1 つの共通鍵のみを復号化し暗号化し直せばよい．

ケーパビリティの配布の場合，送信する際に送信側の共通鍵を用いて復号化し，受信側のセットが保持する受信箱の公開鍵で暗号化され受信箱に保存される．その後，受信箱内の暗号化されたケーパビリティを受信側のセットが保持する秘密鍵で復号化し，そのケーパビリティの共通鍵を作成し保存する．この秘密鍵は，セットが保持する共通鍵で暗号化され保存される．

3.4 制限付きケーパビリティ

ケーパビリティを他のユーザに渡して作業してもらう場合，時間制限や使用回数制限等の制限をつけたいという要望がある．たとえば，明日までに作業を終わらせてもらいたいがそれ以降にはアクセスしてもらいたくないということがある．また，アンケートへの回答の場合，アクセス回数を制限したいことがある．Web で行うテストの場合，期間中に 1 回の受験が原則である．この場合，ケーパビリティに有効期限と使用回数制限をつけることでその要求を満たすことができる．CaStor では，上で述べた有効期限と使用回数制限をケーパビリティに付加することができる．

3.5 ケーパビリティの配布

ケーパビリティの配布には，3.2 節で述べた送信および受信機能を用いる．まず，送信されたケーパビリティは，受信側のセットが保持する受信箱に保存される．その後，そのセットの所有者であるユーザが，受信箱の中から必要なケーパビリティのみを選択しセットに受信し保存する．

配布するケーパビリティには，以下の 2 種類がある．

- **ケーパビリティ**: ただし，ケーパビリティそのものではなく，ケーパビリティへのハードリンクのみである．
- **間接ケーパビリティ**: ケーパビリティを間接的に使用するためのものである．ケーパビリティへのソフトリンクとして扱う．

ハードリンクを送信した場合，受信者はそのハードリンクが指しているケーパビリティの属性である有効期限や使用回数制限等の情報を書き換えることが可能である．しかし，間接ケーパビリティを送信した場合，ソフトリンクであるため受信者は元のケーパビリティの属性を変更することはできない．

ケーパビリティの配布と編集の関係について，図 3 に示す．ユーザ 1 が自分のセット内にあるケーパビリティ A をハードリンクとしてユーザ 2 のセットに送信した場合，ユーザ 2 はハードリンクを使用して元のケーパビリティの属性を編集することができる．しかし，ユーザ 1 がケーパビリティ B のソフトリンクをユーザ 2 のセットに送信した場合，ユーザ 2 は元のケーパビリティの属性を編集することはできずソフトリンクの属性を編集することになる．

CaStor では，ケーパビリティの作成者でなくてもケーパビリティを送信することが可能であるため，権限移譲の際の手間を省くことが可能になる．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渡す．従来，サブ・リーダから一般のメンバにその権限を渡すことはできないが，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ようにする．これにより，リーダの手間を省くことができる．

3.6 ケーパビリティのブラウズ

ケーパビリティを一か所に集中して管理したとしても，大量にあるケーパビリティが関連性なく表示されると管理は複雑になる．CaStor では，次の手法を用

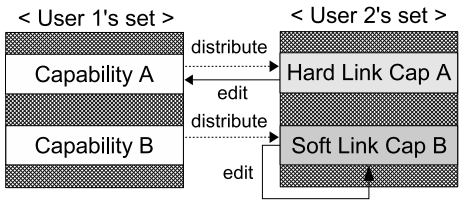

図 3 ケーパビリティの配布と編集

いてケーパビリティをわかりやすく表示する．

- **クラスタリング**: ユーザの命名したケーパビリティ名をタグとして用いてクラスタリングを行う．あるいは，Web 資源の URL をタグとして用いることも考えられる．クラスタリングを行ったケーパビリティを SetNS[14] のモデルを用いて表示することを検討している．SetNS のモデルとは，木構造とは異なり利用する局面に応じて階層の順序を動的に入れ替えることが可能なモデルである．
- **ケーパビリティと間接ケーパビリティ**: ケーパビリティと間接ケーパビリティの見分けがつくように表示する．
- **最終使用時刻**: そのケーパビリティを最後に使用した時刻を表示する．また，その時刻が新しい順にケーパビリティをソートして表示する．最終使用時刻を表示することで，自分の予期しない悪用を察知することができる可能性がある．
- **メタ情報を用いた検索**: 大量のケーパビリティの中から自分の使いたいケーパビリティをケーパビリティ名，Web 資源の URL，あるいは，メモ等のメタ情報を用いて検索する．
- **結合表示**: CaStor では，ユーザは複数のセットを保持することができるため，ログインしているセットのケーパビリティを同じ画面に結合して表示する機能を提供する．この機能がない場合，使用したいケーパビリティが属しているセットにログインを繰り返すことになる．

現在，クラスタリング，ケーパビリティと間接ケーパビリティ，最終使用時刻，そして，結合表示が実装済みである．

4. CaStor の実装

3 で述べた設計に基づいて実装を行っている．本章では，その実装について述べる．

CaStor は，ケーパビリティの管理・配布を行うマネージャ・プログラムと Web 資源に対してアクセスを行うプロキシ[9] から構成される．CaStor のマネージャ・プログラムは，Ruby on Rails[13] を用いて実装している．マネージャ・プログラムでは，データベースとして MySQL を使用している．

4.1 マネージャ・プログラムのデータ構造

CaStor のマネージャ・プログラムは，以下の 3 つのデータ構造を持つ．

- sets: ケーパビリティが属するセットを表す．属性としては，set_ID，セット名，パスワード，set_Key (ケーパビリティを暗号化する共通鍵を暗号化するための鍵)，inbox_ID，inbox_PublicKey (受信したケーパビリティを暗号化するための公開鍵)，inbox_SecretKey (受信箱内のケーパビリティを復号化するための秘密鍵)，および，capability_ID と cap_Key (ケーパビリティを暗号化するための鍵)，あるいは，indirects_ID と ind_Key (間接ケーパビリティを暗号化するための鍵) のペア等がある．
- capabilities: ケーパビリティを保持する．属性としては，capability_ID，ケーパビリティ，ケーパビリティ名，有効期限，および，使用回数制限等がある．
- indirects: 参照元のケーパビリティの capability_ID (capabilities の主キー) と cap_Key を保持する．capability_ID と cap_Key のペアを参照として扱う．属性としては，indirects_ID，capability_ID，cap_Key，ケーパビリティ名，有効期限，および，使用回数制限等がある．
- inboxes: 受け取ったケーパビリティ，あるいは，間接ケーパビリティを一時的に保持する．属性としては，inbox_ID，ケーパビリティ (間接ケーパビリティの場合，cap_ID と cap_Key)，ケーパビリティ名，有効期限，および，使用回数制限等がある．

4.2 暗号化の実装

複数のユーザのケーパビリティを 1 か所に集めて管理しているため，管理者が利用者のケーパビリティを見ることができなように暗号化する．ケーパビリティの暗号化方法としては，乱数を鍵にした共通鍵暗号方式を用いる．

図 4 に暗号化の方法を示す．まず，ケーパビリティ作成時に乱数を生成し，それを cap_Key として使用することでケーパビリティを暗号化する．間接ケーパビリティの場合もケーパビリティと同様に，作成時に乱数を生成し，それを ind_Key として使用することで間接ケーパビリティを暗号化する．cap_Key と ind_Key は，それぞれの ID とともに sets テーブルに保存される．その時，cap_Key と ind_Key は set_Key によっ

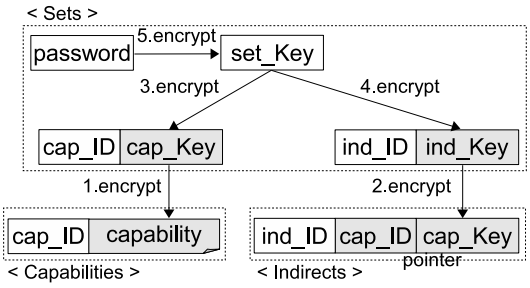

図 4 暗号化

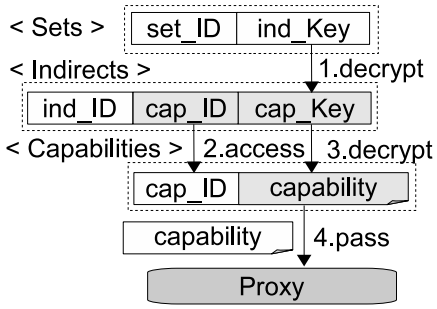

図 5 間接ケーパビリティの使用方法

て暗号化される．set_Key は，そのセットの作成時生成される乱数である．また，set_Key は，そのセットにログインするためのパスワードを用いて暗号化する．

ケーパビリティを使用してある Web 資源にアクセスする場合，まず，ユーザがログインしたセットのパスワードで set_Key を復号化する．次に，使用するケーパビリティを復号化するための cap_Key を set_Key で復号化し，その cap_Key を用いてケーパビリティを復号化する．ユーザが CaStor にアクセスするためのパスワードを変更する場合，CaStor は変更前のパスワードで set_Key を復号化した後，変更後のパスワードで暗号化し直し保存する．set_Key を用いることで，パスワード変更時に全ての cap_Key を復号化して暗号化するという手間を省くことができる．間接ケーパビリティを使用したケーパビリティの間接的な使用に関しては，4.3 節で述べる．また，ケーパビリティの配布時の暗号化・復号化に関しては，4.5 節で述べる．

4.3 間接ケーパビリティの作成と使用方法

間接ケーパビリティを作成する場合，ユーザは元のケーパビリティ，または，間接ケーパビリティのいずれかを選択し保存するセットの set_ID を CaStor に渡す．それらを受け取った CaStor は，capabilities テーブルから cap_ID と cap_Key を取得する．取得した cap_ID と cap_Key を indirects テーブルに保存する．その時に作成された ind_ID と ind_Key を属するセットの sets テーブルに保存する．

間接ケーパビリティを用いてケーパビリティを間接的に使用し Web 資源にアクセスする方法を図 5 に示す．まず，ind_key を用いて cap_ID と cap_key を復号化する．次に，cap_ID を用いて capabilities テーブルに保持されている暗号化されたケーパビリティにアクセスする．その後，cap_Key を用いてそのケーパビリティを復号化し，それをプロキシに渡すことで Web 資源にアクセスする．

4.4 制限付きケーパビリティの実装

現在，有効期限と使用回数制限をケーパビリティにつけることを以下の方法で実装している．

- **有効期限**: まず，有効期限となる時刻を capabilities テーブルに保存する．ユーザがケーパビリティを使用して Web 資源にアクセスする度に現在時刻を取得し有効期限と比較する．有効期限内であればアクセスを許可し，有効期限外であれば拒否する．また，ケーパビリティを表示する時にも現在時刻と有効期限を比較しており，有効期限外であれば「有効期限切れ」と表示するようにしている．
- **使用回数制限**: まず，使用回数の上限を capabilities テーブルに保存する．ユーザがケーパビリティを使用して Web 資源にアクセスする度に使用回数を減らす．ケーパビリティ使用時に使用回数の上限が 0 になっていれば使用を拒否する．また，ケーパビリティを表示する時に上限が 0 になっているケーパビリティには「使用回数切れ」と表示する．

4.5 ケーパビリティの配布の実装

CaStor では，ケーパビリティとケーパビリティへの間接ケーパビリティの 2 つを配布することができる．CaStor では，配布に送信・受信機能を用いる．まず，ケーパビリティの送信時にそのケーパビリティの cap_Key を用いて復号化する．次に，送信先のセットが保持する inbox_PublicKey を用いて暗号化し inboxes に保存する．受信側は，自分の inbox に保存されているケーパビリティの中から必要なケーパビリティ，あるいは，間接ケーパビリティを選択しセットが保持する inbox_SecretKey で復号化する．inbox_SecretKey は，set_Key で暗号化されセットに保存される．復号化された後，cap_Key，あるいは，ind_Key を生成しその鍵を用いて暗号化し capabilities に保存する．最後に，保存したケーパビリティの cap_ID と cap_Key，

あるいは，間接ケーパビリティの ind_ID と ind_Key を sets に保存する．

5. 関 連 研 究

ケーパビリティ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．Hydra では，ケーパビリティはオブジェクトへの参照としてしよされた[17]．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[12]．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[16]．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．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．ただし，CaStor ではシステムの外にある資源を対象とする点が異なる．

パスワード管理アプリケーションとして，2 で述べた MacOS X の Keychain Access[6] がある．Keychain Access は，MacOS X と連携することで OS にログインしたユーザと動的にリンクしている．そのため，ログインしたユーザの保存したパスワードを使用することができる．Keychain Access で保存したパスワードを使用する場合，MacOS X と連携して対応した Web ブラウザに保存されたパスワードを入力する．MacOS X の Keychain Access と比較して，CaStor では，ネットワークに繋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．

シングルサインオンを実現するシステムとプロトコルとしては 2 で述べた Kerberos[15]，Liberty Alliance[5]，そして，OpenID[8] がある．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．Kereros 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時には，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．このような，Kerberos の仕組みはケーパビリティの仕組みに類似しているが，Kerberos の場合，ユーザ認証機能を省略するための機能しかない．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．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．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，Web サイト側が対応することなく使用可能な点で異なる．

外部の Web 資源に認証サービスを提供する API やプロトコルとして Google API[3]，Flichr API[2]，そして，OAuth プロトコル[7] がある．これらは，トークンを使用して保護された 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，その認証情報を元に作成したトークンを外部の Web 資源に渡す．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．これらの API やプロトコルでは，外部の Web 資源にトークンを渡すことが可能であるが，ユーザ間でのやり取りは考慮されていない．これに対して CaStor では，ユーザ間でのアクセス権限を受け渡すことができる．

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・配布を行うサーバ (CaStor) について述べた．CaStor は，ケーパビリティの管理・配布を担当するマネージャ・プログラムと Web 資源へのアクセスを行うプロキシから構成される．ケーパビリティを一元的に管理するため，ユーザの管理の負担を減らす．また，ケーパビリティを他のユーザに安全に配布する機能を提供する．ケーパビリティを配布することにより，アクセス権限の委譲を容易に行うことが可能になる．

今後は，マネージャ・プログラムで実装できていないケーパビリティのブラウズ部分等の実装を行う．現在，3.6 節で述べたように，クラスタリング等のブラウズ方法を考えているが，それ以外に，より管理を容易にするようなブラウズ方法がないか調査する．

参 考 文 献

1) M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian and M. Young : “Mach : A New Kernel Foundation for UNIX Development”, Proceeding of USENIX Summer Conference, pp.93-112, 1986.
2) Flickr Authentication API: “Flickr services: API Documentation”, http://flickr.com/services/api/, 2007.
3) Google API: “Google Code”, http://code.google.com/apis/accounts /Authentication.html, 2007.
4) H.M.Levy: “Capability-Based Computer Systems” , Digital Press, 1984.
5) Liberty Allience Project: “Introduction to the Liberty Alliance”, Identity Architecture Revision 1.0, 2003.
6) Keychain Access: “Introduction to Keychain Services Programming Guide”, 2007.
7) OAuth: “OAuth”, http://oauth.net/, 2007.
8) OpenID: “OpenID”, http://openid.net/developers/, 2007
9) 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, pp.95-102, 2007.
10) M.Mabuchi, Y,Shinjo, A.Sato, and K.Kato: “An Access Control Model for Web-Services that Supports Delegation and Creation of Authority”, The Seventh International Conference on Networking (Accepted for publication), 2008.
11) 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,pp.41-48,2008.
12) S.J.Mullender, G. van Rossum, A.S.Tenenbaum, R. van Renesse, and H. van Staveren: “Amoeba: A Distributed Operating System for the 1990s”, IEEE Computer, Vol. 23, No.5, pp.44-53 (1990).
13) Ruby on Rails: “Ruby on Rails”, http://www.rubyonrails.org/, 2007.
14) 新城 靖, 西尾 克己, 板野 肯三: “SetNS: 記号集合に基づく名前サービス”, 情報処理学会論文誌: コンピューティングシステム, Vol.44, pp.201-214, 2003.
15) J. G. Steiner, B. C. Neuman, and J. I. Schiller: “Kerberos: An Authentication Service for Open Network Systems”, In Proceedings of the Winter 1988 Usenix Conference, pp.191-201, 1988.
16) J.Stichbury and M.Jacobs: “The Accredited Symbian Developer Promer”, Fundamental of Symbian OS, 2006.
17) W. Wulf, E. Cohen, W. Corwin, A. Jones, R.Levin, C.Pierson and F.Pollack: “HYDRA: The Kernel of a Multiprocessor Operating System”, Communication of the ACM, Vol.17, No.6, pp.337-345, 1974.