Web 認証と連携した顧客管理

A Web Authentication Scheme and a Generic Data Model for Application Service Providers

山 田 繁 夫

要 約 本稿では ASP（application service providers）事業者のための顧客管理システムとそれと連携して動作する認証スキームのソフトウェアアーキテクチャについて論じる．ASP 事業者が日々の運営で把握しなければならないデータは多様にあるが，一般的なデータセットを定義することができる．このようなデータセットとして，顧客，利用者，アプリケーションアカウント，利用契約，等のエンティティとそれらの関連を示し，さらに，アプリケーションアカウントの単位でデータを管理する場合に必要な要件を二つ挙げる．Kiban@asaban システムアーキテクチャは，ゲートウェイ，アプリケーション，データベースの 3 層モデルで構成され，それぞれの階層に多様なサービスコンポーネントが存在する．このアーキテクチャでは，ゲートウェイ層はセキュリティポリシーを管理する制御点となり，利用者の認証は認証コンポーネントが担当する．コンポーネントアーキテクチャを採用したことにより，ASP 事業者は自サイトのセキュリティポリシーを満たすように認証方法を選択し，それを動的に構成することができる．

Abstract This paper addresses two areas ; one for a data model for customer database used by ASPs (application service providers) for doing their daily business and the other for an authentication scheme and software architecture using that database. It is required by ASPs to manage their customer related information such as customer profile, service contract and application accounts to be prepared for end user. We introduce a generic data model for customer database. In this model, we identified two requirements in order for ASPs to manage user accounts ; first requirement is for applications to be hosted and second one is for a management system.

Kiban@asaban system architecture consists of three tiers, i. e., gateway, application and database, and each tier contains various service components. The architecture also has an authentication scheme that defines a set of components and interaction between them. In the architecture, gateway is responsible for managing security policy while authenticator component takes care of gathering users credentials to authenticate users. Thanks to component architecture, ASPs are able to choose authentication scheme and configuring into their implementation dynamically.

1. はじめに

ASP（application service provider）の定義には様々なものがあるが[1],[2]，本稿では ASP をインターネット等の広域ネットワークを通じてアプリケーションと関連サービスを個人や企業に提供する事業と捉える．この ASP モデルでは，プログラムやパッケージソフトウェアなどの業務アプリケーション，それらの運用や保守を含めた系を利用者側ではなくデータセンタ側に配置し，利用者は利用契約にもとづきアプリケーションを使用し，その対価を月額料金や年間ライセンス費などの形で ASP 事業者へ支払う．

ASP を提供する事業者が，事業の運営に必要な管理のシステム化を考えるとき，その要件は ASP 事業者ごとに多様であるが，それらおおよそすべての ASP 事業者に共通で一般的な要件と考えられるものをいくつか挙げることができる．第一に，利用契約を締結する顧客とサービスを享受する者――アプリケーションを使用する者――とは，一般的に必ずしも一致しない．たとえば，電子キャビネット ASP を考えるとき，共有キャビネットは複数の利用者で共有され，利用者は契約した顧客により自由に追加や削除がなされる．管理システムはこれら二つを区別できなければならない．第二に，ASP 事業者が複数のアプリケーションをサービスメニューとして提供する場合，どの顧客がどのサービスを利用するかを管理できなければならない．また，利用する権利のない利用者が許可されていないサービスへアクセスすることを禁ずるような制約をビジネスルールとして設けることも考えられるが，この場合，管理システムはこのようなビジネスルールを実装するために必要な情報を維持管理できなければならない．また，管理システムは誰がサービスにアクセスしているのかを決定できなければならない．第三に，管理システムは顧客に請求するサービス利用料金を算出するためにサービス利用実績など算出に必要なデータを採取し，それらを顧客のアカウントへ分配できなければならない．

1999 年，日本ユニシス（以下，当社）は ASP のパイロット事業を開始し，そのとき，多様なサービスメニュに対応できるよう標準化されたプラットフォーム，共通サービス，運用サービスの 3 要素で構成される‘ Kiban@asaban'と呼ぶシステムアーキテクチャを設計した[3]．冒頭の管理システムの要件は顧客管理サービス，課金管理サービス，認証サービスとして共通サービスレイヤやプラットフォームレイヤに実装できる．以下，2 章にて，このシステムアーキテクチャに準拠する顧客管理システムについて考察し，引続き 3 章にて，それにもとづく認証と制約を実現する仕組みを論じる．

2. 顧 客 管 理

2.1 ASP における顧客データモデル

ASP 事業に共通で一般的な顧客管理の論理データモデルを図 1 に示す．図で，アプリケーション（Application）は ASP で稼働するプログラムやパッケージソフトウェアである．サービス（Service）は，そのアプリケーションを使って事業者が提供するサービスメニューであり，顧客が選択する最小単位である．顧客（Customer）は，事業者と利用契約を結ぶ主体であり，企業やビジネスユニット，あるいは個人である．利用契約（Subscription）は事業主と顧客との間でサービスを利用することを合意する契約で，ここでは事業主と顧客の間の n : n の関連としてモデル化している．最終利用者（User）は実際にアプリケーションを使用しサービスを享受する主体で，個人あるいはエージェントのような個人の代理プログラムである．最終利用者は顧客と n : 1 の関係で関連づけられている．

実在するアプリケーションの多くは，アプリケーションアカウント（ApplicationAccount）の概念を持ち，利用者を識別し利用を許可するための仕組みを持っている．モデルではこれをアプリケーションと最終利用者の両者と関連づけられたエンティテ

ィとしてモデル化している．

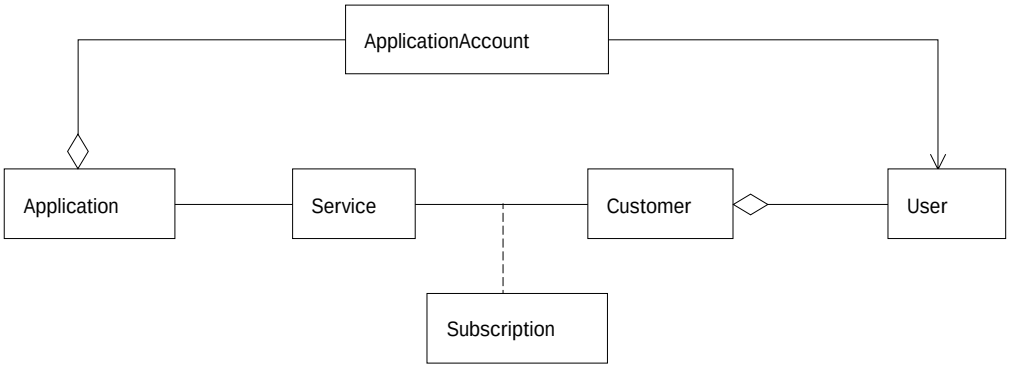

図 1　顧客管理の論理データモデル

2.2　顧客管理システムの設計・実装の考察

図 1 に示した論理データモデルを ER モデルに変換し，それを関係データベースで実装することができる．あるいは，論理モデルを LDAP ディレクトリのスキーマに変換して，それをディレクトリ製品を用いて実装することもできる．関係データベースを使い実装する場合の長所は，一般によく知られているように，複雑な問合せでも SQL 言語を使って効率よく処理可能であること，データベース管理システムがトランザクションの ACID 特性[*1] を面倒みてくれること，などが挙げられる．ディレクトリによる実装の長所は，情報を分散して管理することができ，管理の委譲ができること，それらを結合することでネットワークに広く分散する大規模な情報を扱うことができること，などが挙げられる．

管理システムにて最終利用者やアプリケーションアカウントを管理することを計画する場合には，それを可能とするための，つぎの二つの要件を満たすように設計しなければならない．

1）アプリケーションに対する要件

業務アプリケーションは外部のシステムとアプリケーションアカウントと利用者情報の整合を取るための外部インタフェースを提供すること：

実装される業務アプリケーションがこの要件を満たさないかぎり，管理システムと整合を保つための機構を実装することができず，これらの維持管理を ASP 事業者自身の手による個別の二重管理に委ねざるを得なくなる．大規模なサイトでは最終利用者やアプリケーションアカウントは膨大な数となり，この方法で顧客データの整合性を維持管理することはとても難しくなる．

2）管理システムに対する要件

管理システムは業務アプリケーションとの間でアプリケーションアカウントと利用者情報の整合を取るための方法を規定し，外部インタフェースを規定すること：

残念ながら現時点では，業務アプリケーションが提供すべきそのような外部インタフェースの統一規格は存在せず，現実は，業務アプリケーションが管理する

最終利用者やアプリケーションアカウントと顧客データとの整合を取るために個々の業務アプリケーション毎に個別なインテグレーションを必要としている．

3. Kiban@asaban システムアーキテクチャと認証スキーム

Kiban@asaban システムアーキテクチャ[3]では業務アプリケーションが稼働するアプリケーション層の前段に DMZ（demilitalized zone）とファイヤウォールから構成されるゲートウェイ層がある．利用者からの要求は先ずゲートウェイ層が受取り，アプリケーションゲートウェイ型のファイヤウォールがそれをアプリケーション層へリダイレクトする．この動作は利用者，業務アプリケーションの両者に透過であり，特別に意識されることなく動作する．TCP/IP アーキテクチャ[4]のアプリケーションレベルで動作するファイヤウォールは，利用者を認証し，セキュリティポリシーを適用するチェックポイントを実装する場所となる．

利用者を認証する（利用者が本当に本人であることを確認する）ための仕組みは，共通サービスのコンポーネントのひとつである認証システムが提供する．また，利用者を認証するための情報は，別の共通サービスコンポーネントである顧客管理システムが維持管理する．認証システムは HTTP 認証スキーム等により利用者を認証し，認証に成功するとゲートウェイ層に対して認証トークンを発行する．認証トークンにはユーザ ID 等の情報が含まれており，ファイヤウォールは認証トークンを使ってセキュリティポリシーを適用し，ポリシーにて許可された要求だけをアプリケーション層へリダイレクトする．ファイヤウォール（Firewall），認証システム（Authenticator），顧客管理システム（CustomerDBMS），業務アプリケーション（Application），利用者の使うブラウザや spiders のような検索ロボットなどのユーザエージェント（UserAgent），を機能コンポーネントとしてとらえた場合の，これらのコンポーネントの関係を図 2 に整理する．

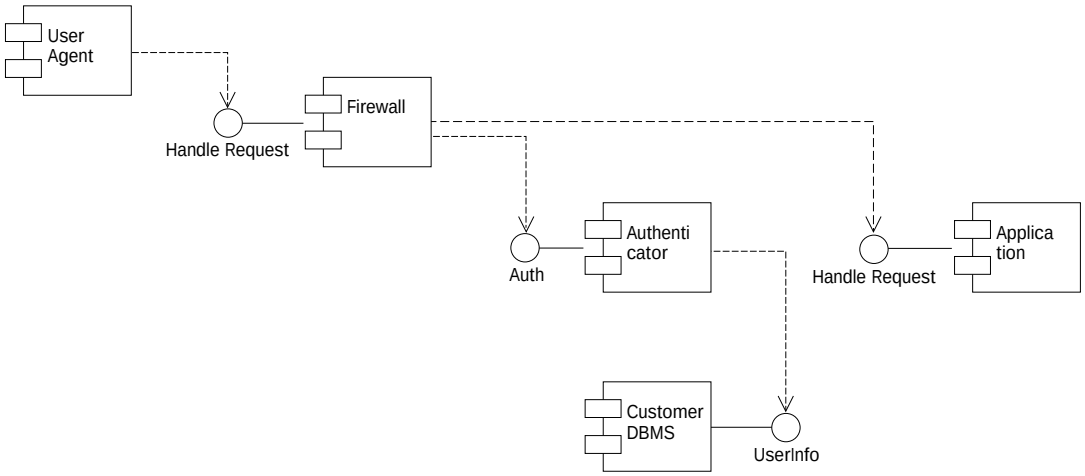

図 2　認証スキームに関連するコンポーネント

認証トークンは利用者側にキャッシュされ，再利用されることを前提にアーキテク

チャが設計されている．認証トークンは有効期限の概念を持ち，有効期限内であれば何回でもトークンを再利用できる．トークンの持つこの特性は，ゲートウェイ層のファイヤウォールが冗長構成をとりながら同時に認証を共有するために重要である．また，複数の ASP サイトで認証トークンを共有すれば，いわゆるシングルサインオンが実現できる．以下，この認証スキームを実装するための認証コンポーネントの要素技術とソフトウェアアーキテクチャを述べる．はじめに一般的な認証コンポーネントについてふれ，ひきつづき具体的な二つの認証コンポーネントについてふれる．

3.1 Generic な認証コンポーネント

まず最初に，すべての認証コンポーネントに共通なコンポーネントの実装技術を考察する．認証コンポーネントは，ファイヤウォールから呼び出され，利用者側のユーザエージェントと共に利用者を認証し，認証に成功すれば認証トークンを発行し，ユーザエージェントの状態を認証済みの状態へ遷移させる．認証済みのユーザエージェントは，アプリケーションへの要求時に常に認証トークンを付加する．

1) ユーザエージェントの状態管理

HTTP 1.1 プロトコル[5]はセッションの概念を持たないプロトコルである．クライアントはサーバへ HTTP リクエストを出し，サーバがそれを処理し結果を HTTP レスポンスとしてクライアントへ返し， そこですべての処理が完結する．サーバはそのリクエストと“その前の”リクエストとを関連づけず，個々のリクエストをばらばらに処理する．プロトコルの持つこの性質から，認証チェックポイントとなるファイヤウォールが HTTP リクエストから利用者を認証できるためには，すべての HTTP リクエストに認証トークンが付与されていることが必要となる．HTTP プロトコルを使って状態を持つセッションを実現する方式として，クッキー[6]を利用する手法や，状態変数を URI（Universal Resource Identifiers）へ埋め込む手法が広く使われている．前提となるアーキテクチャでは，URI は業務アプリケーションが使用し，アプリケーションに認証トークンを URI へ埋め込むことを制約条件として強いることはできないので，クッキーを利用する方法を採用した．認証コンポーネントが認証トークンをクッキーにセットし，ユーザエージェントがそれを HTTP リクエストへ付与し，ファイヤウォールがそれを検査する．

クッキーはユーザエージェント側にキャッシュされるが，永続的にキャッシュされると都合が悪い．クッキーの寿命は認証コンポーネントがコントロールすることができ，認証トークンはユーザエージェントの終了と同時に破棄されるよう設定する．

2) HTTP リダイレクション

はじめに利用者がユーザエージェントを立ち上げた直後は，ユーザエージェントは認証トークンを持たず，認証されていない状態にある．ファイヤウォールは認証されていない HTTP リクエストを受け取ると，ユーザエージェントを認証システムへ誘導する．この時，オリジナルの URI を認証コンポーネントへ伝える必要がある．認証システムは利用者を認証し，認証に成功すればユーザエージェントを認証された状態へ遷移させ，ユーザエージェントを再びオリジナルの

URI へ誘導する．このような誘導は，HTTP リダイレクションとして知られている手法を使い実現する．ファイヤウォールや認証コンポーネントがリクエストを別の URI へ誘導するときには，HTTP リプライのステータスコードに“302 Found”を設定し，Location ヘッダに誘導する先の URI を指定してユーザエージェントへ返す．ユーザエージェントはステータスコードを見て利用者の介入することなく指定された URI へ HTTP リクエストを投げる．

3） 認証トークンの構造設計

ファイヤウォールはすべての HTTP リクエストを検査し，認証トークンが付与されていればその健全性を検証する．典型的な業務アプリケーションでは，1 ページ内に十数のオーダのグラフィクスがページ要素として埋め込まれており，これらのリクエストも全て認証の対象となる．認証トークンを設計するにあたり，トークンを効率よく検証できるような考慮が必要である．また，認証トークンは改竄を検知することができる構造を持たなければならない．

3.2 HTTP 認証スキームを使う認証コンポーネント

一般的な web ブラウザの機能とプロトコルを使って認証コンポーネントを実装する場合には HTTP プロトコル仕様[5]で規定される基本認証方式およびダイジェスト認証方式[7]を利用することができる．HTTP 基本認証は，認証コンポーネント側にあらかじめ登録されたユーザ ID とパスフレーズの組みが HTTP リクエストへ埋め込まれ提示されたものと一致することで本人性を確認する．ダイジェスト認証は，認証コンポーネント側にあらかじめ登録されたユーザ ID とパスフレーズ，および，認証システムがその場で提示するランダムなチャレンジデータを使って互いに同じアルゴリズムでハッシュ値を計算し，利用者が提示した値と認証システムが計算した値が一致することで本人性を確認する．この方式では，パスフレーズがネットワーク上を流れないため，前者に比べて安全である．

これらの仕様をベースに認証コンポーネントを実装する利点は，仕様がデファクト標準であるため，ほぼ全ての web ブラウザがこれを実装しており，利用者側に特別な追加コードがいらないことである．反面，これらいずれの方式も，利用者が，利用者と事業者の二者しか知りえない知識を知っていることにより本人を確認する認証方式であり，利用者側と認証コンポーネント側の両者がどのように安全にその知識――パスフレーズ――を共有できるかという問題を含んでいる．より安全な認証手段を必要とする場合には次節に述べる PKI（public key infrastructure）技術にもとづく認証方式を使うことを検討する必要がある．

HTTP 認証スキームは，認証に必要な情報を HTTP リクエストヘッダを使って運ぶため，インターネットのような解放型ネットワーク上で使う場合には，第三者によるリプレイアタックを防ぐために通信の暗号化と組合せて使用する必要がある．認証に必要な情報を平文で運ぶ場合のリスクに関する議論は文献[7]に詳しい．

3.3 PKI にもとづく第三者認証を使う認証コンポーネント

利用者が ASP 事業者の信頼する認証機関から本人性を証明する鍵証明書を取得し，認証システムが提示するチャレンジデータに本人の秘密鍵を使って署名の上，鍵証明書と共に認証システムへ送り返し，認証システムが鍵証明書にある公開鍵を使って署

名を検証することで本人性を確認することができる．PKI[8]技術にもとづくこの方式は，利用者のみが持ちうるもの――秘密鍵を使って本人を確認する．

一般的な web ブラウザは，SSL プロトコル[9]や TLS プロトコル[10]を使った認証方式を実装しており，これを使って利用者を認証するシステムを実装することができる．両プロトコル共に，はじめに鍵の交換と署名の検証を行うハンドシェイクのフェーズがあり，その後にリクエストとリプライを交換するフェーズへとつづく．ハンドシェイクにて検証された内容をベースに利用者を認証することができる．

証明書の形式は X.509 version 3 公開鍵証明書[11]を使用する．この証明書は，利用者の公開鍵が Subject フィールドで示された持ち主のものであることを証明する．証明書は認証機関による署名が施されており，認証機関が本人確認をする方法――認証局の運用方針――や証明書のプロファイル――認証書の内容――を，認証機関と ASP 事業者との間で事前に合意する．Subject フィールドの値は，利用者本人を一意に指し示す階層的な X.500 ディレクトリでの命名形式となっており，例えば，利用者の国籍，所属組織，氏名の組などで命名される．また，証明書形式にはユーザアカウントや電子メールアドレス等，利用者の属性を格納する場所もあり，これらのいずれを使って本人を確認するかを認証システムが選択する自由度を与える．

X.509 version 3 公開鍵証明書は，証明書に署名をする認証局が階層の概念を持つことができるようになっている．これを利用して，ASP 事業者が信頼する認証局をトラストアンカとして，その下位の認証局らの発行する証明書をすべて有効なものと見なす認証モデルを実現できる．たとえば，e-マーケットプレイスの ASP サイトの場合，トラストアンカをエンタープライズ規模の認証局に設定し，マーケットプレイスに参加するそれぞれの組織が運用する認証局へその組織の証明書発行を委譲するような認証モデルを実装することが可能である．

4. 認証スキームの Apache サーバへの実装

3 章で述べた認証スキームは Apache　HTTP サーバ[12]へ実装することが可能である．以下，Apache HTTP version 1.3 サーバ（以下，Apache）を使って構築する構成例を述べる．例となる構築モデルは，ゲートウェイ層のファイヤウォールと HTTP 基本認証を使う認証コンポーネントで，この他に，顧客管理データベースを格納する DBMS，業務アプリケーションとで構成する．

1)　ファイヤウォール

Apache は HTTP サーバとしての基本機能と，動的に配置可能な機能モジュール群から構成される．図 2 のファイヤウォール機能は，次のようなモジュールの組み合わせで実現することができる．

mod_proxy　ユーザエージェントから受けた HTTP リクエストをアプリケーション層の業務アプリケーションへの要求へ振り向ける．業務アプリケーションが返す HTTP リプライをユーザエージェンへ返す．

mod_SSL　SSL プロトコルと TLS プロトコルをサポートする拡張モジュール

mod_affixe　3 章で述べた認証スキームを実現するためのモジュールで，(1) 認証を受けていない HTTP 要求の認証コンポーネントへのリダイレクシ

ョン，（2）認証トークンの検証，（3）認証トークンからユーザ ID の取り出し，を実装する．

2） **HTTP** 基本認証を使う認証コンポーネント

Apache のコアには HTTP 基本認証とダイジェスト認証サポートする機能を含んでおり，これに認証トークンを発行する機能を付加することで目的の認証コンポーネントを実装することができる．

mod_auth Apache のコアに含まれるモジュールで，HTTP 基本認証とダイジェスト認証サポートする．ユーザ ID とパスフレーズの組をテキストファイルより読み出す仕様なので，最も単純に実装する場合には，定期的にデータベースからユーザ ID とパスフレーズの組を抽出し，テキストファイルを生成し，それを参照するよう構成する．

mod_auth_tds 上記 mod_auth の代替案で，利用者の本人性を確認するため，顧客管理データベースに対してユーザ ID とパスフレーズを問い合わせる．Apache 標準ではなくオープンソースとして配布されているモジュールである．この他にも，類似の機能を持つ利用可能なモジュールがオープンソースの形でコントリビュートされており，ASP サイトのポリシーに合うものを利用する．サイトによっては，セキュリティ要件を満たすために，データベースに格納するパスフレーズは，平文ではなく Apache が提供する MD 5 ダイジェストパスワード値を格納し，受け取ったパスフレーズと比較するようにオリジナルの仕様を変更する必要があるかも知れない．

3） 業務アプリケーション

ファイヤウォールからリダイレクションされたユーザの HTTP リクエストには，認証コンポーネントにより認証された利用者名が HTTP リクエストの Authorization ヘッダに付加され，CGI や servlet 等で実装されるアプリケーションはそれを標準インタフェースを使って取り出すことができる．2 章で述べた二つの要件を満たすように設計されたアプリケーションの場合，アプリケーションアカウントと利用者名は同値となるので，アプリケーションによるサインオンプロセスを省くことができる．

5. おわりに

本稿では，ASP 事業者が事業の運営に必要な顧客管理システムの要件と，認証システムについて述べた．ASP サイトの規模が大きくなり，稼働する業務アプリケーションの数が多くなってくると，ユーザへの権限付与をより小さな粒で集中管理するセキュリティポリシーマネージャの機能が運営に必要な要件となる．このようなセキュリティポリシーマネージャは，セキュリティポリシー，（認証を受けた）利用者の情報，サービスへのポインタ，を入力として許可あるいは不許可を返すコンポーネントとして実装し，ファイヤウォールと結合させることができる．サービスに名前を与える命名規則，利用者に名前を与える命名規則，ポリシー記述言語，サービス利用のためのトークンの設計，等，さらに検討すべきテーマも残されているが，近い将来，

この機能が Kiban@asaban システムアーキテクチャへ実装されるだろう．

今まで，ASP サービスの利用者がヒトであることを前提に議論を進めてきたが，今後は最近議論されている XML　Web サービスフレームワーク（Web サービス）をベースとするリモートプロシジャコール型の ASP サービスが登場すると考える．本稿で述べた認証スキームは Web サービスとの親和性が高く，Web サービス型 ASP のための ASP 基盤への拡張と共に，実装されるだろう．

＊ 1 （ACID 特性）原子性 Atomicity, 一貫性 Consistency, 隔離性 Isolation, 永続性 Durability

参考文献
［1］ ASP Industry Consortium, <URL : http : //www. aspindustry. org>
［2］ ASP Industry Consortium Japan, <URL : http : //www. aspicjapan. org>
［3］ 保科剛, ASP サービス基盤「Kiban@asaban」, ユニシス技報 68 号, Vol. 20 No. 4, 2001 年 3 月, pp. 135–143
［4］ J. Franks, P. Hallam–Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, “ HTTP Authentication : Basic and Digest Access Authentication ”, RFC 2617, June 1999.
［5］ Postel, J.,“ Transmission Control Protocol ”, STD 7, RFC 793, September 1981.
［6］ R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners–Lee, “ Hypertext Transfer Protocol–HTTP/1.1 ”, RFC 2616, June 1999.
［7］ D. Kristol, L. Montulli,“ HTTP State Management Mechanism ”, RFC 2965, October 2000.
［8］ ITU–T Recommendation X. 509, “ The Directory–Authentication Framework ”, 1988
［9］ Kaliski, Burton S., Jr.,“ An Overview of the PKCS Standards ”, An RSA Laboratories Technical Note, November 1, 1993.
［10］ A. Frier, P. Karlton, and P. Kocher,“ The SSL 3.0 Protocol ”, Netscape Communications Corp., Nov 18, 1996.
［11］ T. Dierks, C. Allen.,“ The TLS Protocol Version 1.0 ”, RFC 2246, January 1999.
［12］ The Apache Group, <URL : http : //httpd. apache. org>

執筆者紹介　山　田　繁　夫（Shigeo Yamada）
1983 年日本ユニシス(株)入社．人工知能/オブジェクト指向モデリング/インターネット技術の研究，アプリケーション開発に従事．現在 asaban.com 事業部に所属．