権限証明書と SSL 相互認証による匿名アクセス制御方式

梅澤 健太郎[†,☆] 齋藤 孝道[††] 奥乃 博[†††]

現在，SSL により安全な通信路の確保と通信相手の認証によるアクセス制御を行う方法がよく利用されている．しかし，この方法は，サーバが管理する ACL とクライアントの ID に基づく「任意アクセス制御方式」であるので，サーバが ACL を管理するコストが生じたり，匿名アクセス制御の実現が難しいといった問題が生じる．このような問題に対する解決策として，本論文では，X.509 証明書を権限証明書として利用した SSL 相互認証モードによる「匿名アクセス制御方式」を提案し，Web サーバのアクセス制御に適用する．本方式は，クライアントとして通常の Web ブラウザが使用できるので，利便性が高いだけでなく，本論文で提案する SSL 相互認証モードは，既存の SSL と同様に通信路の暗号化が可能である．また，一般的な SSL サーバ認証モードの利用方法では問題となるクライアントの無制限なサーバへのアクセスを避けることができる．さらに，クライアントは本方式により，サーバに対して匿名での権限行使が可能となる．

A Privacy-enhanced SSL Access Control with Authorization Certificates

Kentaro Umesawa,[†,☆] Takamichi Saito[††]
and Hiroshi G. Okuno[†††]

An access controll using SSL is widely used to authenticate each other and provide a secure connection. Since it is based on a kind of DAC using server's ACL and a client's ID, DAC causes several problems: difficulty of realizing an anonymous access control, and an expensive cost of handling ACL. In this paper, to cope with these problems, we present an *anonymous* access control scheme that uses X.509 certificates as authorization certificates and the SSL mutual authentification mode. The proposed anonymous access control is applied to Web server access control. The resulting system has the following advantages: (1) High usability because no special browser is not needed. (2) Improved security due to a secure connection provided by SSL. (3) Avoidance of unrestricted accesses to the server. (4) Anonymous access to the server.

1. はじめに

近年，インターネットの普及にともない，安全かつ簡潔なクライアントのアクセス制御機構に対する期待が増しつつある．現在，WWW（World Wide Web）では SSL（Secure Sockets Layer）[4] を用いたアクセス制御が広く行われている．SSL を用いることにより通信相手の認証，通信路の機密性・完全性の確保が可能とされている．SSL では，公開鍵と ID（識別子）の対応を保証する X.509 証明書を利用する．さらに，サーバ・クライアントモデルという観点では，Web サーバと Web ブラウザを利用したアクセス制御を行うことができる．この場合，まずサーバはクライアントを認証し，その ID を確定する．次にサーバは ID と権限の対応を記した ACL（Access Control List）を，クライアントの ID を検索キーとして参照し認可を行う．現行の SSL によるアクセス制御も含め，上記のようなサーバにおける ACL を用いたアクセス制御方式は，一般に任意アクセス制御と呼ばれる．この方式は，以下の問題がある：

(1) サーバが認証と認可を行うので，サーバはアクセス制御対象のクライアントに関する ACL を管理する必要がある．
(2) サーバはつねにクライアントを認証する必要

† 東京理科大学大学院理工学研究科情報科学専攻
Graduate School of Sciences and Engineering, Science University of Tokyo
†† 東京工科大学工学部情報工学科
Department of Information Sciences, Tokyo University of Technology
††† 京都大学大学院情報学研究科知能情報学専攻知能メディア講座 音声メディア分野
Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University
☆ 現在，株式会社東芝研究開発センター
Presently with TOSHIBA R&D Center

がある．そのためクライアントの匿名性が損失する．

(3) クライアントが複数のサーバに権限を行使する場合，それぞれのサーバに ID とある程度の個人情報（たとえば年齢，性別）を与える必要がある．そのためクライアントの個人情報が漏洩する機会が増加する．サーバはクライアントが登録した個人情報により，クライアントの権限を決定し，ACL を作成する．

アクセス制御において，認証を行い，利用者が誰であるかを知る必要がないこともある．状況によっては，サーバはクライアントがどのような権限を持つかどうかを単に確かめれば，アクセス制御を行うことはできる．特に，匿名アクセスを許可したいときには，認証をせずにアクセス制御だけを行いたい．本論文では，このような要求を満たすアクセス制御方式を実現する．

現在，上記の問題を解決する方式として，SPKI（Simple Public Key Infrastructure）[2),3)]をベースとする Web サーバの匿名アクセス制御方式[5),6)]がある．しかし，このシステムでは，ユーザは Web ブラウザ以外に別途ソフトウェアを必要とする．一方，SSL は広く普及しており，ユーザは一般的な Web ブラウザだけで SSL を利用することができる．本論文では，この SSL の利便性と文献 5)，6) の方式による利点を両立した匿名アクセス制御を提案する．すなわち本方式は，X.509 証明書により実現した**権限証明書**（後述）を，SSL 相互認証モードでクライアントの証明書として用いることで，サーバにおいて ACL を必要としないアクセス制御を行う．この方式では，新たな主体である **Issuging Agent**（後述）が権限証明書を発行することで，サーバに対するクライアントの匿名性を確保している．さらに，クライアントは 1 つの IssuingAgent を利用することで，複数のサーバに対して権限を行使できる利点もある．本方式を適用した Web サーバでは，サーバを管理するコストを軽減し，クライアントのプライバシの重視が可能な匿名アクセス制御が実現できる．ここでプライバシの重視とは匿名性の確保と個人情報漏洩の機会の減少を指す．ただし，TCP/IP ベースの通信において，IP アドレスが個人情報にかかわる状況もある．その際には，Proxy サーバを利用したりすることで，自分の通報の発信 IP アドレスを隠す通信を想定する．また，SSL はアプリケーション層のプロトコルに依存しない．そのため本方式は FTP（File Transfer Protocol），telnet といった他のプロトコルにも適用可能である．

2. 準　　備

2.1 表 記 方 法

本節では，本論文を通して用いる表記を説明する．主体 X の ID と対応づけされた公開鍵，秘密鍵は P_X，S_X と表す．さらに証明書の内容を「$\langle$」と「$\rangle$」でくくって表現し，その証明書に秘密鍵 S_X で電子署名が施されていることを，$\langle \ldots \rangle_{S_X}$ と表現する．また，MD5[13)]などのハッシュ関数によるデータ X のハッシュ値を $H(X)$ と示す．さらに n 枚の X.509 証明書から構成される証明書パスを $\{crt_1, crt_2, \cdots, crt_n\}$ と示す．ここで，crt_1 は自己署名されたルート証明書であり，階層型の構造を持つ CA から構成される公開鍵基盤においては，ルート CA の証明書に相当する．また，crt_n はこの証明書パスの利用者の証明書（以後，エンドエンティティ証明書）となる．さらに，主体 X が，秘密鍵 S_X を用いた署名によって「主体 Y の ID と公開鍵 P_Y との対応」を保証する有効期限 V の X.509 証明書を $Crt_{ID}Y = \langle X, Y, P_Y, V \rangle_{S_X}$ のように表記し，Y の ID 証明書と呼ぶ．また ‘Server’ と ‘Client’ は，提案するシステムを構成する各主体（後述）を示し，一般のサーバ・クライアントモデルにおけるそれらと区別できるように，前者を英字で，後者をカタカナでそれぞれ表記する．

2.2 SSL の概要と再考

SSL は，暗号化・認証の機能を提供し，安全性を高めた通信を実現する[11)]．SSL は Alert Protocol，Record Layer Protocol，Handshake Protocol から構成されている．特にサーバ S とクライアント C の暗号通信に利用する共通鍵の交換は，Handshake Protocol で行う．さらに，Handshake Protocol では C が S を認証する（サーバ認証），また S が C を認証する（クライアント認証）ことが可能であり，サーバ認証を行う **SSL サーバ認証モード**，サーバ認証とクライアント認証の両方を行う **SSL 相互認証モード** の 2 つの利用形態がある．通信相手の認証は，通信相手から送信される証明書パスのエンドエンティティ証明書に含まれる公開鍵を利用する．エンドエンティティ証明書自体の正当性は，証明書パスを検証することで確認する（2.2.1 項参照）．また，以下で示すように「サーバ認証と通信の暗号化」,「クライアント認証と任意アクセス制御」はそれぞれ密接な関係を持っている．

2.2.1 SSL における証明書パスの検証

現在，一般的な Web ブラウザには PSE（Personal

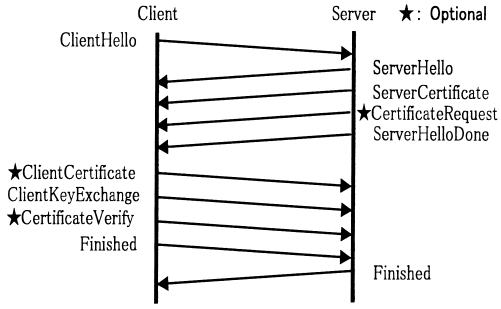

図 1 Handshake Protocol でのメッセージ交換
Fig. 1 Exchanged messages at Handshake Protocol.

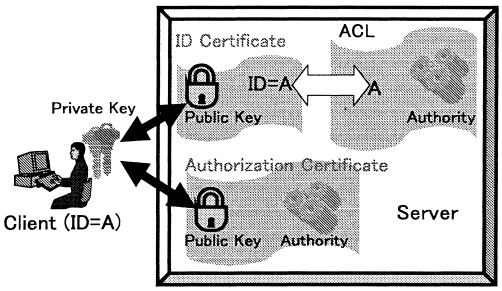

図 2 クライアント認証における要素の対応
Fig. 2 Binding of elements on client authentication.

Secure Environment）☆があり，ユーザはそこにいくつかの CA の証明書を trust-point☆☆として保管する．SSL のクライアント認証を行う Web サーバは，trust-point をあらかじめ設定☆☆☆する．そして SSL におけるクライアントあるいはサーバは，証明書パスをそれぞれの trust-point を用いて検証する．具体的な検証作業の内容は，RFC2459[1)]に従う．

2.2.2 サーバ認証と通信の暗号化の関係

サーバ認証では，**図 1** の ServerCertificate で S が C に送信した証明書パスを利用する．まず，C は証明書パスの検証を行いエンドエンティティ証明書 $Crt_{ID}S$（以後，S のサーバ証明書）の正当性を確認する．次に C は P_S を利用して，自分が作成した Pre Master Secret（やりとりされるデータの機密性・完全性の保護に用いる共通鍵の元となるデータ）を暗号化し，図 1 の ClientKeyExchange として S に送信する．S はこれを秘密鍵 S_S により復号し，Pre Master Secret を得る．ここで，S 以外の主体は秘密鍵 S_S を利用できないため，C は Pre Master Secret を共有できた相手が，$Crt_{ID}S$ で示された S であると確認できる．これがサーバ認証である．また，C と S は共有した Pre Master Secret から生成した共通鍵を利用して，機密性・完全性を保ちつつデータの送受信を行う．

2.2.3 クライアント認証と任意アクセス制御の関係

クライアント認証では，図 1 の ClientCertificate で C が S に送信した証明書パスを利用する．まず，S は証明書パスの検証を行いエンドエンティティ証明書 $Crt_{ID}C$（以後，C のクライアント証明書）の正当性を確認する．次に，C は秘密鍵 S_C を用いて電子署名を付加したメッセージを，図 1 の CertificateVerify として S に送信する．そして，S は正当性を確認した $Crt_{ID}C$ に含まれる P_C を利用して，電子署名を検証する．SSL におけるクライアント認証とは，サーバ S が，クライアント証明書に含まれる公開鍵に対応する秘密鍵を通信相手が所持していることを確認する作業である．ここでは，C 以外の主体は秘密鍵 S_C を利用できない．そのため，S は電子署名を施した相手が，$Crt_{ID}C$ で示された C であると確認できる．これにより S は確定した C の ID を検索キーとして ACL を参照し，C の権限を確定することができる．以上のようにして，ID 証明書を用いたクライアント認証と ID に基づくアクセス制御機構を用いてアクセス制御を行う．

3. システムの構成と利用環境

3.1 システムの要件

現行の SSL によるアクセス制御では，前述のとおりサーバは ACL を管理する必要がある，クライアントはプライバシを損なう可能性がある，などの問題がある．これらを解決する本システムの要件を示す：

(1) **Server（後述）は，ACL を管理することなく Client（後述）のアクセス制御ができる**：本方式による SSL 相互認証モードでは，クライアント証明書として権限と公開鍵の対応を保証する権限証明書（後述）を利用する．そのため Server は，SSL のクライアント認証の手続きに成功した Client が権限証明書に記述された権限を持つことを確認できる（**図 2**）．そして，その権限証明書を直接利用してアクセス制御を行う．すなわち，Server は Client の ID を利用して権限を決定する必要がなく，ACL を必要としない．このことにより，Server はアクセスしてくる Client の名前空間や個人情報を管理する必要がない．したがって，Server が Client の個人情報を管理する必要がないことから，Server 自身が個人情報の漏洩元となることもなくなるので，その際に生じる責任問題といったリス

☆ 自身が保持する証明書および秘密鍵，直接的に信頼される CA の証明書などを保存する信頼できるローカル記憶域．
☆☆ ユーザが直接信頼する CA の証明書．
☆☆☆ mod_ssl では，Apache のコンフィグレーションファイルで指示子 SSLCACertficateFile で指定する．

クを回避できるといった副次的な利点も存在する．
また，アクセス管理ポリシーを記述する主体と，アクセス管理ポリシーを参照して ACL を作成する主体が同一である必要はないため，本方式においては，リソースの管理者である Server がアクセス管理ポリシーの「記述」を行い，Client の名前空間や個人情報を管理する IA が記述されたアクセス管理ポリシーを「参照」して ACL（後述の $ListIA_2$ に相当する）を作成・管理する．このような「記述」と「参照」の分離の実現により，各主体が管理する範囲を縮小している．

(2) **Client は，Server に匿名でアクセスできる**：本方式では，Client は Server など Issuing Agent 以外の主体に対して匿名性を保持できる．権限証明書は Client の ID を直接含まないため，Client を認証し権限証明書を発行した Issuing Agent のみが権限証明書と Client の ID の対応を知りうるからである．ただし，本論文での匿名性は Issuing Agent と Server の結託がないことを前提としており，暗号プロトコルの分野でいわれる匿名性とは種類が異なるものであることに注意する．

(3) **既存のシステム．特に Client のシステムを極力変更しない**：権限証明書を X.509 証明書により実現して，既存の PKIX（PKI with X.509）対応のソフトウェアで構成でき，さらに Client は SSL の機能が備えられた Web ブラウザ以外必要としないシステムを目指す．すなわち，現在一般的に利用されている SSL によるアクセス制御を活用する．
これにより，サービスを提供する側は専用のクライアントソフトウェアを用意する必要はなく，Client は Web ブラウザのみで複数のサービスを利用できる．一方で，本方式のような匿名アクセス制御を専用のクライアントソフトウェアを導入することで実現する場合，サービスを提供する側は専用のクライアントソフトウェアを用意する必要があり，Client は利用するサービスの数に比例した専用のクライアントソフトウェアを必要とすることになるだろう．

(4) **Client の個人情報流出の機会を減少させる**：Client は 1 つの Issuing Agent に個人情報を登録することで，その Issuing Agent と提携する複数の Server に対して権限の行使ができる．複数の Server それぞれに対して個人情報を提示する必要がないため，結果として Client の個人情報が流出する機会が減少する．

3.2 システムを構成する主体

提案システムは 4 つの主体から構成される：

(1) **Issuing Agent（以降，適宜 IA と表記する）**：Server からの委託を受け，この主体に登録された管理下にある Client を認証し，権限証明書（後述）を発行する主体．IA は特定のサービスを提供するための存在であり，公の存在である CA とは異なることに注意する．たとえば，ISP（Internet Service Provider）などに相当する．また，複数の Server から委託を受ける可能性がある．Client に対する権限証明書の発行機構で構成される．権限証明書の発行機構は，Client の認証機構と，権限証明書の作成機構からなる．

(2) **Server**：権限証明書の発行権限を IA に委譲し，Client がクライアント証明書として提出した権限証明書を用いて SSL 相互認証モードによるアクセス制御を行う主体．ただし，Server 自身は，どの Client がアクセスしてきたかを問題とはせず，偽造された権限証明書の使用などによる損失を防ぐことができればよい．信頼する IA を利用することで，自身は不特定多数の Client の名前空間や個人情報を管理することなく，Client を選別しサービスの提供を行う．Client のアクセス制御機構と権限証明書の失効管理機構から構成される．

(3) **Client**：IA から権限証明書を取得し，その証明書を SSL 相互認証モードのクライアント証明書として Server に提出することによってサービスを享受する主体．

(4) **CA**：ID 証明書を発行する主体で，RFC2459[1] で規定される主体を想定する．IA，Server のサーバ証明書を発行する．たとえば，商用サービスとしてサーバ証明書を発行する機関に相当する．

3.3 証明書について

3.3.1 権限証明書

権限証明書とは，公開鍵と権限の結びつきをその発行者に対する信頼の下で保証する公開鍵証明書である．本方式では，以下の要素を格納した X.509 証明書に発行者の電子署名を付加したものとする：

$Issuer$：この権限証明書を発行した主体の持つ権限と対応する公開鍵のハッシュ値．

$Subject$：$Authority$ と対応する公開鍵のハッシュ値．

$PublicKey$：$Authority$ と対応する公開鍵．

$Delegation$：ブール値．True，または，False．$Subject$がさらに権限を委譲可能か示す．

$Authority$：権限を示す．

$Validity$：証明書の有効期間を示す．

そして，権限 $Au2$ に関する権限証明書 $AuCrt(Au2)$ を以下のように表記する：

$$\langle H(P(Au1)), H(P(Au2)), P(Au2), D, Au2, V\rangle_{S(Au1)}$$

ここで，公開鍵 $P(Au1)$ がこの権限証明書を発行した

主体の持つ権限 $Au1$ と対応しており，公開鍵 $P(Au2)$ が権限 $Au2$ と対応している．この証明書は，$S(Au1)$ の保持者が，$S(Au2)$ の保持者に対して権限 $Au2$ を有効期限 V で保証するものである．さらに，$S(Au2)$ の保持者の権限委譲の可否は D で示されている．

権限証明書の証明書パスを検証する場合は，2.2.1 項で示した検証作業に加え，権限委譲の検証を行う．権限委譲の検証とは，証明書パスに含まれるエンドエンティティ証明書以外の証明書の $Delegation$ が True であることを確認する作業である．

3.3.2 システムで利用する X.509 証明書

システムで利用する X.509 証明書を以下に示す：

- Server S のサーバ証明書（ID 証明書）：
  $Crt_{ID}S = \langle CA, S, P_S, V_1 \rangle_{S_{CA}}$
- IssuingAgent IA のサーバ証明書（ID 証明書）：
  $Crt_{ID}IA = \langle CA, IA, P_{IA}, V_2 \rangle_{S_{CA}}$
- S，IA の ID 証明書を発行する CA の ID 証明書：
  $Crt_{ID}CA = \langle CA, CA, P_{CA}, V_3 \rangle_{S_{CA}}$
- S が S 自身に対して発行する権限証明書．$P(Au1)$ に対応する $S(Au1)$ は S が保持する．このことから，権限 $Au1$ は S の保持する権限を示す：
  $AuCrt(Au1) =$
  $\langle H(P(Au1)), H(P(Au1)), P(Au1), \mathrm{True}, Au1, V_4 \rangle_{S(Au1)}$
- S が IA に対して発行する権限証明書．$P(Au2)$ に対応する $S(Au2)$ は IA が保持する．このことから，権限 $Au2$ は S が IA に与える権限を示す：
  $AuCrt(Au2) =$
  $\langle H(P(Au1)), H(P(Au2)), P(Au2), \mathrm{True}, Au2, V_5 \rangle_{S(Au1)}$
- IA が C に発行する権限証明書．$P(Au3)$ に対応する $S(Au3)$ は C が保持する．このことから，権限 $Au3$ は IA が C に与える権限を示す：
  $AuCrt(Au3) =$
  $\langle H(P(Au2)), H(P(Au3)), P(Au3), \mathrm{False}, Au3, V_6 \rangle_{S(Au2)}$

3.4 主体の保持する情報

システムを運用するうえで，Server，IssuingAgent は以下のような情報を保持する：

(1) Server S の保持する情報：
  (a) $ListS_1$：有効期間内に失効した権限証明書 $AuCrt(Au3)$ に含まれるシリアル番号とその証明書の $Issuer$ の値 $H(P(Au2))$ の組のリスト．
  (b) $ListS_2$：有効期間内に失効した権限証明書 $AuCrt(Au2)$ に含まれるシリアル番号のリスト．

(2) IssuingAgent IA の保持する情報：
  (a) $ListIA_1$：C の ID と C に発行した権限証明書 $AuCrt(Au3)$ のシリアル番号 $Serial$ の組のリスト．問題の発生時に管理上必要とあらば，IA は $Serial$ をキーにしてこのリストを検索し，C を特定する．
  (b) $ListIA_2$：C の ID，権限，権限の有効期間の組をエントリとするリスト．
  (c) $ListIA_3$：C の ID，パスワードのリスト．

3.5 システムにおける主体間の通信

ここでは，主体間の通信について説明する．

(1) Server S と IssuingAgent IA との間の通信：今回の実装では，この通信はオフラインを用いる．しかし，S と IA はそれぞれ 3.3.2 項で述べた証明書 $Crt_{ID}S$，$Crt_{ID}IA$ を保持しており，これらを ID 証明書として用いることで SSL の通信を行うことも可能である．

(2) Client C と IssuingAgent IA との間の通信：$Crt_{ID}IA$ をサーバ証明書とする SSL サーバ認証モードを用いることで，C はサーバ認証を行うことができる．また，IA は C を SSL の通信路上でパスワードにより認証する．C にも $Crt_{ID}C$ を持たせることで，証明書による相互認証を行うことも可能である．

(3) Server S と Client C との間の通信：本論文での提案方式であり，$Crt_{ID}S$ をサーバ証明書とし，$AuCrt(Au3)$ をクライアント証明書とする SSL 相互認証モードの通信路である．以下で図 1 に沿った処理の流れを示す：
  (a) S は，クライアント証明書の trust-point として $AuCrt(Au1)$ を設定する．
  (b) S は，C に ServerCertificate としてサーバ証明書 $Crt_{ID}S$ とそれを発行した CA の証明書 $Crt_{ID}CA$ からなる証明書パスを送信する．
  (c) C は，この証明書パスを検証する．
  (d) C は，S に ClientCertificate として証明書パス $\{AuCrt(Au1), AuCrt(Au2), AuCrt(Au3)\}$ を送信する．
  (e) S は，この証明書パスを検証する．
  (f) S と C は，ClientKeyExchange により 2.2.2 項のように Pre Master Secret の

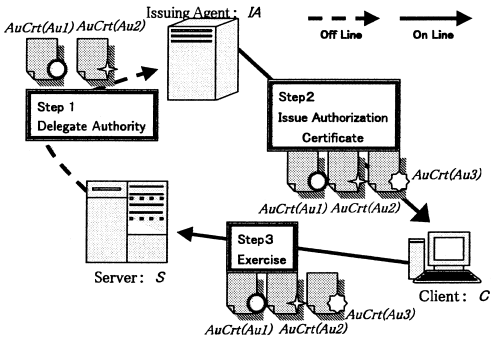

図 3 権限証明書の発行から利用までの処理の概要図
Fig. 3 Overview of processing steps.

共有を行う．

(g) C は S に，CertificateVerify として $S(Au3)$ による電子署名を付加したデータを送信する．

これらの処理の結果，**(f) によりサーバ認証は正しく行われ，C は S を認証し，認証した S と機密性・完全性の確保されたデータの送受信が可能となる．また，S は (g) で送信された電子署名を $P(Au3)$ で検証することで，通信相手が権限証明書に記された権限を持つことを確かめることができる**．

3.6 システムの処理概要

各主体は 3.4 節で示した情報を保持し，3.5 節で説明した通信路を用いて処理を行う．また各証明書は 3.3.2 項で定義したものとする．

Server S は証明書 $AuCrt(Au1)$，$AuCrt(Au2)$ を作成し，IssuingAgent IA に $AuCrt(Au1)$，$AuCrt(Au2)$ および $S(Au2)$ を送信することで，「Client に権限証明書を発行する権限」を委譲する．これを発行権限委譲フェーズと呼び，**図 3** の Step1 に対応する（実装における詳細は 4.3.1 項）．この際の S と IA の通信は，3.5 節 (1) で説明した．

次に，Client C は IA から権限証明書 $AuCrt(Au3)$ の発行を受ける．まず，IA は C を認証し，そして $AuCrt(Au3)$ を作成する．次に，C は $\{AuCrt(Au1), AuCrt(Au2), AuCrt(Au3)\}$ を IA から取得する．これを権限証明書発行フェーズと呼び，図 3 の Step2 に対応する（実装における詳細は 4.3.2 項）．この通信路は 3.5 節 (2) で説明したように，SSL で暗号化される．

最後に，C は S に $AuCrt(Au3)$ を含む証明書パス $\{AuCrt(Au1), AuCrt(Au2), AuCrt(Au3)\}$ を提出し，自分の持つ権限 Au を行使する．ここで S は，クライアント認証により通信相手が $S(Au3)$ を保持することを確認する．クライアント認証が成功した場合，S は C の証明書パスの権限 $Au1$，$Au2$，$Au3$ を縮約して C の権限 Au を決定し，C に Au に対応するサービスを提供する．これを権限行使フェーズと呼び，図 3 の Step3 に対応する（実装における詳細は 4.3.3 項）．この処理は 3.5 節 (3) で説明したように，SSL で暗号化される．

3.7 権限証明書の失効管理

権限証明書 $AuCrt(Au1)$，$AuCrt(Au2)$，$AuCrt(Au3)$ の失効管理を示す．

3.7.1 $AuCrt(Au1)$ の失効管理

Server は，新たな $AuCrt(Au1)$ を作成し，それを trust-point として設定する．そして，失効した古い $AuCrt(Au1)$ をルート証明書とする証明書パスをクライアント証明書として受け付けない．

3.7.2 $AuCrt(Au2)$ の失効管理

IA がオフライン（$Crt_{ID}S$ を利用した通信路上で，行うことも可能である）で，S に $AuCrt(Au2)$ のシリアル番号を通知する．そして，S は $ListS_2$ に通知されたシリアル番号を記録する．

3.7.3 $AuCrt(Au3)$ の失効管理

C は S に $AuCrt(Au3)$ のシリアル番号 $Serial$ を通知する．この処理は $AuCrt(Au3)$ を利用して 3.5 節 (3) で説明した通信路で行い，S は C が $AuCrt(Au3)$ の公開鍵 $P(Au3)$ に対応する秘密鍵 $S(Au3)$ の保持者であることを確認する．そして，S は $ListS_1$ に，「$Serial$ と $AuCrt(Au3)$ の $Issuer$ の値 $H(P(Au2))$」の組を記録する．

3.7.4 失効した権限証明書の検証

失効した権限証明書の検証を行う場合，Server とそれ以外の主体では検証方法が異なる：

(1) Server S の検証方法：
  (a) $AuCrt(Au1)$：S にとってこの証明書は自分が発行したものであるため，この証明書の有効性は自身で判断できる．
  (b) $AuCrt(Au2)$：$ListS_2$ に，$AuCrt(Au2)$ のシリアル番号が存在しないことを確かめる．
  (c) $AuCrt(Au3)$：$ListS_1$ を参照して，「$AuCrt(Au3)$ のシリアル番号と $AuCrt(Au3)$ の $Issuer$ の値 $H(P(Au2))$ の組」が存在しないことを確かめる．

(2) S 以外の検証方法：$Crt_{ID}S$ を利用した SSL サーバ認証モードの通信路で，S に問合せを行う．

4. 実装システム

提案方式を適用した Web サーバの匿名アクセス制御システムについて，各主体の実装方法の一例，X.509 証明書による権限証明書の実現方法の一例を示す．そして，この実装システムで想定される運用例を示しつつ，その処理概要を示す．

本実装は本論文の枠組みに基づく Web サーバの匿名アクセス制御の実現例の 1 つであり，本論文の枠組みに基づいた他のサービスを提供する際の 1 つの指針となる．しかし，この実装はあくまで提案方式の実装の一例を示すものであるため，本実装がすべての Web ブラウザで動作することを保証するものではなく，また，将来にわたって動作を保証するものではないことに注意する．

4.1 システムを構成する主体

登場する主体の本実装での実現方法について記述する：

(1) IA について：Apache1.3.19 [14] を OpenSSL 0.9.6 [15]☆および mod_ssl2.8.3-1.3.19 [16]☆☆を利用して SSL 対応にした Web サーバとして実現する．Client の認証機構は，$ListIA_3$ をパスワードファイルとする Apache の基本パスワード認証機構を用いる．権限証明書の作成機構は，Perl のプログラムである．このプログラムでは，Client から公開鍵 $P(Au3)$ を受信し，OpenSSL を利用して $AuCrt(Au3)$ を作成する．

(2) Server について：Apache1.3.19 を OpenSSL 0.9.6 および mod_ssl2.8.3-1.3.19 を利用して SSL 対応にした Web サーバである．さらに Perl で作成したモジュールを利用するために mod_perl1.2.5 [17]☆☆☆も利用する．アクセス制御機構は，Perl で作成したプログラムを用いる．このプログラムは，権限委譲の検証（4.2.1 項参照），権限の縮約（4.2.3 項参照）および証明書失効の検証（3.7.4 項 (1) (a)，(b) を参照）の処理を行い，クライアント証明書の検証とその権限に応じたアクセス制御を行う．権限証明書の失効管理機構は Perl で作成したプログラムで，3.7.3 項の $ListS_1$ への記録処理，さらには 3.7.4 項 (2) で説明した応答処理を行う．

(3) Client について：Web ブラウザを利用する．このブラウザは公開鍵ペアを生成する機能を持ち，SSLVersion2.0 以降をサポートし，さらにユーザごとの PSE を持つものとする．実装では，Netscape Communicator 4.76 [18] を利用した．

(4) CA について：簡便のため OpenSSL0.9.6 のコマンドラインツールを用いて実現し，$Crt_{ID}CA$, $Crt_{ID}S$, $Crt_{ID}IA$ を作成するプログラムを用意する．

4.2 実装システムでの権限証明書

X.509v3 証明書による権限証明書の実現方法の一例を示す．シリアル番号と 3.3.1 項で示した要素を X.509v3 の以下のフィールドに格納する．ここで，この証明書を発行する権限と対応する公開鍵を $P(AuX)$，この証明書の権限と対応する公開鍵を $P(AuY)$ とする（具体例を付録に示す）：

- serialNumber：通常の X.509 証明書と同様に，各発行者にとってユニークなシリアル番号を格納する．この値と issuer フィールドの値の組を証明書の失効管理に利用する．
- issuer：通常は，証明書の発行主体の DN（Distinguished Name）を格納する．本方式では，3.3.1 項の $Issuer$ を格納する．具体的には，バイナリデータである $H(P(AuX))$ を Base64 符号化し，その文字列を CommonName の部分に格納する．ここで，実装において CommonName にはサイズ制限があるため，公開鍵を Base64 符号化した文字列を直接利用しない．
- subject：通常は，証明書の被発行者の DN を格納する．本方式では，3.3.1 項の $Subject$ を格納するために利用し，$H(P(AuY))$ を $Issuer$ と同様に Base64 符号化し格納する．
- subjectPublicKeyInfo：通常は，subject フィールドの DN で識別される主体と対応する公開鍵を格納する．本方式では，3.3.1 項の $PublicKey$ を格納する．すなわち，公開鍵 $P(AuY)$ を格納する．この公開鍵は，NetscapeComment フィールドに記述された権限と対応している．
- validity：通常の X.509 証明書と同様に，この証明書の有効期限（3.3.1 項の $Validity$）を格納する．
- basicConstraints：通常は，この証明書の subject フィールドで示される主体が，CA であるかをブール値で表す．本方式では，3.3.1 項の $Delegation$ を格納する．そして，subjectPublicKeyInfo フィールドの公開鍵 $H(P(AuY))$ に対応する秘密鍵 $H(S(AuY))$ の持ち主が，権限委譲を行うために権限証明書を発行することが可能かをブール値で示す．
- NetscapeComment：通常は，任意のコメントを格納する場合に利用する．本方式では，3.3.1 項の $Authority$ を格納し，この権限証明書で保証

☆ 暗号ライブラリの集合．SSL 通信用ライブラリも含む．
☆☆ Apache を HTTPS に対応させるモジュール．
☆☆☆ Apache に Perl インタプリタを組み込むためのモジュール．

される権限を文字列として格納する．実装での権限の記述形式は 4.2.2 項で示す．

この節で示した X.509 証明書を利用した権限証明書の実現方法は，あくまで一例を示すものであるので，X.509 証明書の他のフィールドを利用した実現方法もある．たとえば，本論文の実装例では NetscapeComment を利用して格納している権限情報を，X.509V3 の証明書拡張に格納することも可能である．

4.2.1 権限証明書の証明書パスの検証

本実装では，Server S による権限証明書の証明書パスの検証は，以下の 2 つを行う：

(1) 2.2.1 項で説明した検証作業：クライアント認証において mod_ssl が行う．

(2) 権限委譲の検証：エンドエンティティ証明書以外の証明書パスの証明書について，basicConstraints フィールドが Ture であることを確認する作業で，S のアクセス制御機構が行う．

4.2.2 権限の記述様式

本実装における権限は，Web サーバへのアクセス権限である．本実装では，Web サーバに対する権限空間は，リクエストメソッド☆の集合と処理対象の URL の集合との積集合で構成されると定めた．そして，あるリクエストメソッドが許可される場合，その許可対象となる URL を列挙する．拡張 BNF 記法で示す：

⟨Authority⟩::=⟨AuthPerMethod⟩
| ⟨Authority⟩,⟨AuthPerMethod⟩
⟨AuthPerMethod⟩::=⟨RequestMethod⟩:⟨URLSet⟩
⟨RequestMethod⟩::=GET|POST|HEAD
⟨URLSet⟩::=URL|URL ⟨URLSet⟩

ここで，URL は具体的にアクセス制御の対象となる URL（Uniform Resource Locator）である．また ALL という予約語を用意し，この予約語で Server が自身に発行する権限証明書 $AuCrt(Au1)$ の権限を示す．以下に例を示す．Client C が，index.html，some.jpg に対する GET，index.cgi に対する POST の権限を持つ場合，C の権限を

GET:index.html some.jpg,POST:index.cgi

のように記述する．本実装では上記のように権限の記述様式を定義したが，権限は Server が解釈できればよいのでシステムに応じて様々に記述できる．

4.2.3 権限の縮約

複数枚の権限証明書の表す権限を確定する際には，それらの権限を縮約する必要がある．本実装で，2 つの権限 $Au1$，$Au2$ を縮約する場合，それらの共通部分をとる．以下に具体例を示す．

$Au1$=GET:URL_1 URL_2,POST:URL_3
$Au2$=GET:URL_1 URL_4 $Au3$=ALL

のとき，縮約した権限 $Au=Au1 \cap Au2$，$Au'=Au2 \cap Au3$ は次のようになる：

Au=GET:URL_1 Au'=GET:URL_1 URL_4

4.3 具体的な運用例とその処理の流れ

本方式の実装システムは，様々な形で運用される．ここでは例として，「ISP であり Client の名前空間や個人情報などの管理業務を行う IA」と「Web コンテンツに対する匿名アクセス制御を低コストで行いたい S」さらに「IA の管理下にあり，S の提供するサービスを S に匿名で享受したい C」さらに「社会的に信頼できる第三者機関が運営する CA」から構成されるシステムの運用を考える．この運用例においては以下のような前提を設ける：

(1) C は IA に，権限証明書の発行を受ける際の認証に用いる ID とパスワードをあらかじめ登録してある．

(2) C は IA に，自分の個人情報（年齢，性別，etc）をあらかじめ登録してある．

(3) C は，S，IA にサーバ証明書を発行した CA の証明書 $Crt_{ID}CA$ を trust-point として Web ブラウザに保持している．

そして，S は事前に IA に「どのような条件を満たすクライアントに対して，どの程度の権限（サービス）を与えるか」というアクセス管理ポリシーを記述している．IA はこの情報を参照し，自身が持つ Client の情報とあわせて $ListIA_2$ を作成する．以下に，この運用例における実装システムの処理概要をフェーズごとに示す．各フェーズは 3.6 節に示したものである．

4.3.1 発行権限委譲フェーズ

S は IA に「Client に権限証明書を発行する権限」を委譲する．この際の S と IA の通信は，3.5 節 (1) で示した．S は，Perl で生成したプログラムを用いて $AuCrt(Au1)$，$AuCrt(Au2)$ を作成する．ここで，$AuCrt(Au1)$ および $AuCrt(Au2)$ の $Validity$ および $Authority$ は，IA と S の事前の合意に従って決定されている．そして，IA に $AuCrt(Au1)$，$AuCrt(Au2)$ および $S(Au2)$ を送信する．ここでは例として，S は URL_1，URL_2，URL_3，URL_4 に関する権限を IA に委譲するとし，$AuCrt(Au1)$ の権限 $Au1$，$AuCrt(Au2)$ の権限 $Au2$ は以下とする：

$Au1$=ALL
$Au2$=GET:URL_1 URL_2 URL_3,POST:URL_4

☆ Web サーバへのクライアントの処理要求の内容を示す文字列．

4.3.2 権限証明書発行フェーズ

C は IA から権限証明書 $AuCrt(Au3)$ を取得する．この通信路は 3.5 節 (2) で述べたように，SSL で暗号化される．

権限証明書の作成機構へのアクセス

C は，IA に Web ブラウザでアクセスし，Web ブラウザにより公開鍵ペア $P(Au3)$，$S(Au3)$ を生成後，$P(Au3)$ を IA に送信する．

認証機構による認証

IA は，権限証明書の作成機構にアクセスがあると，認証機構である Apache のパスワード認証を行う．パスワード認証の成否は，$ListIA_3$ により判断する．

権限証明書の作成機構による権限証明書の作成

IA の権限証明書の作成機構は，まず認証機構が行った認証で確定した C の ID をキーとして $ListIA_2$ を検索し，C の権限 $Au3$ と有効期間 V_6 を決定する．ここで例として，$Au3$=GET:URL_1,POST:URL_4 とする．また，C が送信したデータから $P(Au3)$ を取り出す．本方式ではクライアント間の権限委譲は考えないため，権限委譲の値は False とする．以上の情報から IA の権限証明書の作成機構は，$AuCrt(Au3)$ を作成する．そして，$AuCrt(Au3)$ のシリアル番号と C の ID との対応を $ListIA_1$ に記録し，$AuCrt(Au1)$，$AuCrt(Au2)$，$AuCrt(Au3)$ を C に提示する．

権限証明書の取得

C は IA が作成した $AuCrt(Au3)$，$AuCrt(Au2)$，$AuCrt(Au1)$ をダウンロードし，Web ブラウザにインポートする．

4.3.3 権限行使フェーズ

C は S に $AuCrt(Au3)$ を含む証明書パス $\{AuCrt(Au1), AuCrt(Au2), AuCrt(Au3)\}$ を提出し，自分の持つ権限を行使する．この通信路は 3.5 節 (3) で示したように，SSL で暗号化される．

アクセス制御機構へのアクセス

C は，S がアクセスを制限している URL にアクセスする．ここでは例として URL_4 にメソッド POST でアクセスしたとする．

SSL Handshake Protocol

S と C は Handshake Protocol を行う．この処理は 3.5 節 (3) で示した．S は mod_ssl により C の証明書パスに対して 2.2.1 項で説明した検証処理を行う．

アクセス制御機構による証明書の検証

S のアクセス制御機構は，4.2.1 項 (2) の権限委譲の検証処理を，証明書パスの $AuCrt(Au1)$，$AuCrt(Au2)$ に対して行う．次に，3.7.4 項 (1) (a)，(b) の証明書失効の検証を行う．最後に，4.2.3 項に従い C の権限 Au $(= Au1 \cap Au2 \cap Au3)$ を算出し，C のアクセスの成否を判断する．この例では，Au=GET:URL_1,POST:URL_4 となるため，C は URL_4 に対して POST でアクセスが可能と判断される．

5. 議　　論

本提案方式に関する議論を行う．本方式の安全性の議論は，文献 5)〜7) における議論と同様であるため紙面の都合上割愛する．

5.1 実装システムの評価

Server，IA，Client を CPU：モバイル Celeron 600 MHz，メモリ：128 MB，OS：VineLinux2.1.5 の計算機を用いて実現して本実装の評価を行った．

IA の権限証明書の作成にかかる平均処理時間は 147 msec であった．ただし，この数値は IA における権限証明書の作成機構自体の処理時間を測定したもので，Web ブラウザによる鍵生成の時間は含めていない．

また，Server の権限証明書の検証にかかる平均処理時間は 41 msec であった．ただし，この数値は Server におけるアクセス制御機構自体の処理時間を測定したものであり，通常の SSL 相互認証の処理は含めていない．

5.2 運用上の問題とその影響範囲について

本提案方式は，IA，Client，Server の間で証明書のやりとりが閉じた枠組みであり，サービスにかかわる当事者同士が枠組みを利用する際の取り決めを共有することで，運用上の混乱を避けることが可能である．また，本方式により実現されたサービスに何らかの問題が生じても，影響はそのサービスに限られたものであり，サービスに参加していない主体に影響が広がることはない．

5.3 サーバ証明書について

本実装では，IA，Server のサーバ証明書としては ID 証明書を利用することで，サーバ証明書の失効ごとに生じる設定更新の頻度を減少させる．これは一般的に権限の生存期間は ID のそれよりも短く，よって証明書の有効期間も権限証明書の方が短いことを考慮している．ただしこのことは，本論文の主旨であるサーバによるクライアントの匿名アクセス制御を損なうものではない．

5.4 匿名性について

提案システムで利用する権限証明書に含まれる公開鍵は，それぞれ権限ごとに異なったものとすることを前提とする．そのため，Server S は Client C のある権限 $Au1$ とそれとは別の権限 $Au2$ にそれぞれ対応

づけられた 2 つの公開鍵から，2 つの権限を有する主体が同一の主体であることは知りえない．しかし，C が S に対して同一の権限証明書を複数回提示するときに，それに関連づけられた同一の公開鍵から，S が同一の主体からのアクセスであることを推測できるという追跡性の問題がある．この問題は，1 回の使用を前提とした使い捨ての公開鍵を利用することで解決できるので，本方式で非追跡性は実現可能である．また，本質的に，システム全体の観点では認証を行っているので，「誰だか分からない」という意味での匿名アクセスではないことに注意されたい．

5.5 現行の仕様および運用方式への提言

本論文では，IA という存在を導入することで Client の名前空間や個人情報の管理を Server から分離している．IA は，CA のような誰もが信頼すべき公的な存在ではなく，特定のサービスに特化した主体であり，Server から与えられたアクセス制御ポリシーに従って権限証明書を発行する．このような主体を利用することで，サーバは自身が管理すべき情報を減らすことができ，PKI を利用したサービスを容易に提供できる．また，X.509 証明書に権限を格納するフィールドを標準として定義し，その仕様を定めることで PKIX のフレームワークが，権限管理なども含めた汎用的なものとなるだろう．

5.6 関 連 研 究

本方式と同様，安全な通信路の確保と権限証明書によるアクセス制御の汎用的な枠組みを提案する研究として文献 9) がある．しかし，この方式ではクライアントとサーバの両方が SPKI ベースのソフトウェアを新たに準備・利用する必要があり，既存のソフトウェアだけで実現可能な本方式と比較して利便性に劣る．

また本方式では，クライアントは個人情報をあらかじめ IA に登録し，その個人情報を権限証明書に含める権限を決定する際に利用する．それに対して，クライアントがサーバに匿名で権限を行使する際に，個人情報を権限証明書とともにサーバに提示し，それらの情報からサーバがアクセス制御を行う方式も提案されている[8),10)]．どちらの方式を利用するかは，実現するアプリケーションの性質によるところが大きい．

6. ま と め

本論文では，SSL 相互認証モードと権限証明書を利用して SSL の新たな利用方法を提案し，Web サーバの匿名アクセス制御を実現した．このシステムは，PKIX ベースで設計された既存のフリーソフトウェア（Apache，OpenSSL，etc）といくつかの Perl プログラムから構成されている．また，クライアントは Web ブラウザ以外のソフトウェアを必要としない．このシステムにより以下が達成できる：

(1) **サーバの管理コスト減**：サーバ（システムにおける Server）はクライアント（システムにおける Client）の名前空間や個人情報を管理せずに済む．

(2) **匿名性**：クライアントはサーバに対して匿名でアクセスできる．

(3) **個人情報漏洩の機会の減少**：クライアントは，1 つの IA に登録することで，複数のサーバに権限を行使できるようなシステムを実現できる．

参 考 文 献

1) Housley, R., et al.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC2459 (1999).
2) Ellison, C.: SPKI Requirements, RFC2692 (Sep. 1999).
3) Ellison, C., et al.: SPKI Certificate Theory, RFC2693 (May 1999).
4) Freier, A.O., Karlton, P. and Kocher, P.C.: The SSL Protocol Version 3.0 (Nov. 1996).
5) 齋藤孝道，梅澤健太郎，奥乃 博：プライバシーを重視するアクセス制御システムの一方式，電子情報通信学会論文誌 D-I，Vol.J84, No.11, pp.1553–1562 (2001).
6) Saito, T., Umesawa, K. and Okuno, H.G.: An Access Control with Handling Private Information, *Proc. 15th International Parallel and Distributed Processing Symposium 2001*, ISBN 0-7695-0990-8, IEEE Computer Society (2001).
7) 梅澤健太郎，齋藤孝道，武田正之，奥乃 博：SPKI によるプライバシー重視のアクセス制御の PKIX を利用した実現，コンピュータセキュリティシンポジウム 2001, pp.331–336 (2001).
8) 梅澤健太郎，齋藤孝道，奥乃 博：プライバシーを重視したアクセス制御機構の提案，情報処理学会論文誌，Vol.42, No.8, pp.2067–2076 (2001).
9) Howell, J. and Kotz, D.: End-to-end authorization, *Proc. 4th Symposium on Operating Systems Design and Implementation*, pp.151–164 (2000).
10) 本城信輔，洲崎誠一：プライバシに配慮した個人情報による WWW 認証・アクセス制御，コンピュータセキュリティシンポジウム 2001 (Oct. 2001).
11) 齋藤孝道，萩谷昌己，溝口文雄：公開鍵を用いた認証プロトコルについて，情報処理学会論文誌，Vol.42, No.8 (2001).
12) CCITT. Recommendation X.500: The directory-overview of concepts, models and services (1988).

13) Rivest, R.: The MD5 message-digest algorithm, RFC1321 (1992).
14) http://www.apache.org/
15) http://www.openssl.org/
16) http://www.modssl.org/
17) http://perl.apache.org/
18) http://www.netscape.com/

付録　実装における $AuCrt(Au3)$

以下は実装での $AuCrt(Au3)$ を OpenSSL によりテキスト表示したものである（紙面の都合上空白などの整形をしている）：

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 23 (0x17)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: CN=NzJjMDhlNMmE2ZWY0YTI3OWE4YWI0ZmIK
    Validity
      Not Before: Mar 21 20:24:08 2002 GMT
      Not After : Mar 21 20:24:08 2003 GMT
    Subject: CN=MmZmMGE5OThkNTE5MmI1ZjM2Yjk3YzMK
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:e3:67:71:fc:bb:22:09:6c:09:d9:fa:
          ...................
          ba:57:ca:f9:83:32:46:d3:6b
        Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Subject Alternative Name:
        <EMPTY>
      X509v3 Basic Constraints:
        CA:FALSE
      Netscape Comment:
        GET:index.html ipsj2001.pdf
Signature Algorithm: md5WithRSAEncryption
  68:d3:17:50:c1:63:88:93:45:05:53:bf:96:cf:
      ..............................
  87:e7
```

(平成 13 年 12 月 4 日受付)
(平成 14 年 6 月 4 日採録)

梅澤健太郎（正会員）

1976 年生．2002 年東京理科大学大学院理工学研究科情報科学専攻修士課程修了．同年（株）東芝入社．現在，（株）東芝研究開発センターに勤務．2000 年度第 62 回情報処理学会全国大会大会奨励賞受賞．第 17 回電気通信普及財団テレコムシステム技術学生賞受賞．

齋藤　孝道

1998 年東京理科大学大学院理工学研究科情報科学専攻修士課程修了．現在，東京工科大学工学部情報工学科講師．博士（工学）．

奥乃　　博（正会員）

1972 年東京大学教養学部基礎科学科卒業．日本電信電話公社，NTT，科学技術振興事業団北野共生システムプロジェクト，東京理科大学を経て，2001 年 4 月より京都大学大学院情報学研究科知能情報学専攻教授．博士（工学）．この間，スタンフォード大学客員研究員，東京大学工学部客員助教授．音環境理解，音楽情報処理，推論機構等の研究に従事．1990 年度人工知能学会論文賞，IEA/AIE-2001 最優秀論文賞，第 17 回電気通信普及財団賞奨励賞等受賞．本学会英文図書出版委員．著編書：『インターネット活用術』（岩波書店，1996），“Computational Auditory Scene Analysis”（共編，LEA，1998），“Advanced Lisp Technology”（共編，Taylor and Francis，2002）等．