第XII部

公開鍵証明書を用いた利用者認証技術

第 12 部
公開鍵証明書を用いた利用者認証技術

第 1 章　moCA WG の概要

PKI(Public Key Infrastructure) は、暗号鍵の所有者を確認する手段を提供する基盤技術である。信頼された第三者である CA(Certification Authority) は非対称鍵暗号方式の公開鍵と所有者名に電子署名を施すことで、この組合せを証明する。PKI を通信相手の認証に用いる場合、認証を行う者は CA による組合せの証明を検証することになる。

PKI は元来 ITU-T X.509[13] において定義された技術である。IETF において実用的な標準化活動が行われ、様々なアプリケーションで実装された。

その結果、電子メールにおけるメッセージ認証と暗号化を実現する S/MIME やコネクション型プロトコル TLS(Transport Layer Security) で X.509 証明書フォーマットの公開鍵証明書 (以降証明書と呼ぶ) を利用することができるようになった。

moCA WG は CA の振る舞いや証明書の扱いに注目し、moCA (members oriented CA) の運用実験を行ってきた。しかし証明書を扱うことができるアプリケーションが次々に開発されるに従い、サーバやクライアントがいかに証明書を扱うかの議論を行う必要が出てきた。この状況をかんがみて、moCA WG では CA 証明書の有効期限を延ばして管理負荷を軽減し、また、WIDE メンバに対する証明書 (WIDE メンバ証明書) の発行手順を毎年検討してきた。今年度の moCA WG の活動や実験は、WIDE メンバ証明書と証明書を扱うアプリケーション、更に WIDE メンバ証明書の内容に基づく応用的な用途に注目して行われた。

本章では、今年度の moCA WG の活動は “CA の実験運用”、“WIDE メンバー証明書”、“サーバ証明書”、“証明書を応用するシステムの利用” の 4 つに分類される。

1.1 CA の実験運用

moCA と呼ばれる実験用の CA は ICAP(ICAT CA Package) を利用して運用されており、証明書要求の受け付け、証明書の発行、CRL の発行、証明書の検索などを行うことができる。moCA は、moCA 自身の実験運用だけでなく、ICAP の改良、moCA が発行する証明書をアプリケーションで利用するといった目的のためにも運用されている。

今年度 moCA は WIDE メンバ証明書・サーバ証明書の二種類の証明書を発行した。WIDE メンバ証明書は、これまでの要求を行ったユーザに発行していたクライアント証明書の扱いを変更し、WIDE メンバ全員に発行されるものである。サーバ証明書は、要求を行ったサーバ管理者に発行するものである。

次節以降では、この二種類の証明書の背景と使われ方について概説するとともに、証明書のフィールドに格納されている値の変化が、今年度の moCA WG の活動にどのように影響したかについて述べる。

1.2 WIDE メンバ証明書

WIDE メンバ証明書は名前が示す通り、WIDE メンバ自身が WIDE メンバであることを示すために使われる。WIDE メンバは、これを合宿申込み Web サーバクライアント認証やメンバ同士でやりとりするメール (S/MIME) を利用することができる。

これまで、このような用途で使われる証明書は、moCA に証明書要求を送信したユーザにのみ発行されていた。またユーザ自身が鍵対の生成を行って moCA に証明書要求を送信することで、ユーザ自身が秘密鍵を管理していた。

しかし、鍵対の生成と証明書要求の送信を行うことができるクライアントソフトウェア (Web ブラウザ) は Netscape Navigator と Internet Explorer に限られている。またこれらの Web ブラウザが送信する証明書要求を受け付けるインターフェースは、共通していない。そのための Web ブラウザごとに用意する必要があった。更に、証明書要求を行うユーザは、WIDE メンバの中でも少数であるため、アプリケーションの利用実験を行う際に、多くのユーザ

の利用を見込むことができなかった。

そこでユーザ自身によって鍵対を作成してもらう方針を変更し、moCA が鍵対の生成と証明書の発行、WIDE メンバへの配付を行うことにした。この方法で発行された証明書が WIDE メンバ証明書である。WIDE メンバ証明書の配付を実現するには、moCA の既存の機能に加え、ユーザがクライアントソフトウェアに秘密鍵と公開鍵をインポートできる形式である PKCS#12 に変換する機能、WIDE メンバに送信する機能などを実現する必要があった。この実装については 2.3 節で述べる。

WIDE メンバ証明書の発行によって、WIDE メンバ証明書の発行 (2002 年 6 月 18 日) 以前のクライアント証明書の合計のユーザ数が 93 であったのに対し、WIDE メンバ証明書発行以降 (2003 年 1 月 30 日現在) は、726 となった。WIDE メンバ証明書の発行を受けたユーザが必ずクライアント証明書を利用することは想定できないが、鍵対の生成機能や証明書要求を送信する機能がないソフトウェアのユーザが、クライアント証明書を利用できる状況を作ったと考えることができる。

1.3 サーバ証明書

サーバ証明書は https サーバで使用されることを想定して発行されている。moCA が発行したサーバ証明書は、2002 年 3 月以降の WIDE 合宿の申込み Web サーバ (e-side 社の協力による) や、two WG の Web サーバなどで使用されている。

サーバ証明書は、WIDE メンバ証明書などのユーザ向けの証明書を利用しているユーザであれば誰でも発行の要求を行うことができる。

1. サーバ証明書を利用したい WIDE メンバは、moCA ML か CA オペレータに対して、その旨のメールを送信する。
2. CA オペレータは、S/MIME の署名などを用いて WIDE メンバかどうかの確認を行い、moCA のサーバ証明書発行インターフェースを用いて発行する。

サーバ証明書の発行手順はサーバ証明書の発行申請 [177] で説明されている。

サーバ証明書を利用した https サーバは、多くの商用 Web サーバで使われており、サーバ認証を行うことができる Web ブラウザも数多く開発されている。WIDE におけるサーバ証明書利用する場合には、ユーザの Web ブラウザに WIDE ROOT CA をルート CA とする WIDE 系列の CA 証明書をインストールする手間が必要である。しかし、WIDE 合宿の申込み Web サーバの利用等に際して、繰り返し説明を続けてきたせいか、WIDE におけるサーバ証明書利用の是非を問う場面は少なくなってきている。今後、Web ブラウザの TLS のサポート状況に合わせるなどの理由で、合宿申し込みサーバで http サーバが使われることはないと考えられる。

1.4 PKI を使う認証システムの利用及び開発活動

今年度新たに発行した WIDE メンバ証明書のフィールドの値は、サーバがユーザを識別することを容易にすることを考慮して決定された。moCA が WIDE メンバ証明書以前に発行していたクライアント証明書は DN(DistinguishedName) 値として、WIDE メンバの氏名をローマ字で記述した文字列を格納していた。しかしこの形式では、クライアント証明書のフィールドの値を元にデータベースを利用する際に不都合が生じる。氏名をローマ字で記述した文字列を検索キーとして利用すると、大文字や小文字の違い、同姓同名のユーザの識別、スペースの個数の違いなどを考慮しなければならない。

そこで WIDE メンバ証明書の DN には、WIDE 番号と氏名をスペース一つで区切った文字列を格納することにした (表 2.1 参照)。WIDE 番号を証明書の DN に格納しておくことで、WIDE 番号を利用したデータベースの検索が可能になる。

またクライアント認証を行うサーバが、証明書に格納されている値を外部プログラムに渡すことができると、その値を使った別の処理ができるようになる。例えば、認証されたユーザのメールアドレスを記録し、特定のページにアクセスしたユーザのメールアドレスの一覧を作成することができる。その他に、あるユーザが置いたファイルを、そのユーザだけが削除できるようにするなど、アクセス制御に利用することが考えられる。次章以降ではそれぞれの活動の詳細について述べるとともに、IETF における相互運用実験の報告や WIDE Root CA の活動について述べ、このような特徴を生かした応用に関しては、3 章にて述べる。

第2章 WIDE メンバ証明書の配付実験

本章では、WIDE メンバ証明書の配付実験について、一般の証明書配付との関係、検討内容、実装について述べる。

2.1 証明書発行と、CA・RA との関係について

2.1.1 証明書発行手順のパターン

CA を構築すると決めた後の一般的な悩みどころの一つは、ユーザに対して証明書を発行し配付する手順である。IETF で RFC 化された PKI 関連プロトコルは多数あるが、特定の証明書発行手順までは標準化されておらず、下記のポイントから各自で手順を決める必要がある。

- 鍵 (秘密鍵と公開鍵) を誰が作成するか (ユーザ or RA or CA)
- 発行された証明書をオンラインで配付するかオフラインで配付するか

ここ数年の商用 CA サービスや様々な組織での CA 構築状況を見ると、おおむね以下の 2 種類が典型的である。

(a) ユーザが鍵作成、かつ、オンライン配付型

ユーザが (コマンド実行によって) 鍵を作成し、ユーザから証明書発行を RA または CA にオンラインで要求し、RA または CA からの証明書の配付をオンライン (電子メールや WWW など) で受け取る。

(b) RA が鍵作成、かつ、オフライン配付型

RA がユーザの鍵を作成し、RA から証明書発行をオンラインで要求し、ユーザは RA からの秘密鍵と証明書の配付をオフライン (郵送、手渡しなど) で受ける。CA が RA を兼ねる場合もこの方法に含める。

上記 2 種類ほど典型的ではないが、下記の方法もある。

(c) RA が鍵作成、かつ、オンライン配付型

RA がユーザの鍵を作成し、RA から証明書発行をオンラインで要求し、ユーザは RA または CA からの秘密鍵と証明書の配付をオンラインで受け取る。

(a) は、電子メールソフトや VPN 装置などの製品に適用されている手続きであり、商用 CA サービスと連携した製品では典型的である。

(b) は、ユーザの鍵を IC カードなどに格納して配付する際に適用されている手続きであり、企業 PKI などのプライベート CA では典型的である。

(c) は、想定する応用場面が複数あって、応用ターゲットの鍵作成や証明書発行要求機能にバラツキがある場合や、応用ターゲットが鍵作成機能を持っていない場合を考慮した手続きである。

2.1.2 証明書発行時の本人確認パターン

証明書発行手順の中で重要な点は、発行対象となっていない人の証明書発行、あるいは証明書を受け取るべきでない人への配付を防止することであり、何らかの本人確認が必要である。(a) はユーザが証明書発行要求を行うため、証明書発行要求時のなりすましに注意する必要がある。これに対し、(b),(c) は誰に証明書を発行すべきかを RA が把握しているため証明書発行時の不正の可能性は (a) より少ないはずであるが、証明書配付時のなりすましに注意する必要がある。

(a) は証明書に対応する秘密鍵をユーザが持っているかどうかの確認を行え、でたらめな鍵に対する証明書発行を防ぐことができる。鍵を作成出来なければ証明書発行要求時のなりすましは成功しないし、CA 側に証明書発行の記録が残るため、なりすまされたことは検出できる。これに比べると、(c) のオンラインによる証明書配付時のなりすましの方はユーザ本人や CA 側に気づかれにくい分脅威は増す。しかし、実運用という観点で捉えれば、PKI 以外の従来の認証方法の初期登録と比べて悪くならないケースもあり得る。

具体的な本人確認の例について以下に挙げる。

(a) では、以下の 3 つを挙げる。

- あらかじめ本人確認に必要なパスフレーズ (製品を購入した人だけがわかるパスフレーズなど) を CA・RA から配っておいて、証明書発行要求時に確認する
- 証明書発行要求とともに、本人確認に必要な書類を別途ユーザにオフラインで要求し確認する
- 証明書配付時に、電子メールなどによる到達性確認をもって本人確認に代える

(b) では、

- 証明書の配付時に確実に本人に届くようにする

(c) では、以下の 4 つを挙げる。

- 既存の本人確認手段を利用して、証明書配付時に、本人だけがアクセス可能な場所から配付する
- 既存の本人確認手段を利用して、証明書配付時に、本人だけが復号可能となるようにして配付する
- あらかじめ本人確認に必要なパスフレーズを CA・RA から配っておいて、証明書配付時に、本人だけがアクセス可能な場所に証明書を置く
- あらかじめ本人確認に必要なパスフレーズを CA・RA から配っておいて、証明書配付時に、本人だけが復号可能となるようにして配付する

2.2 WIDE における証明書発行の検討

2.2.1 WIDE における情報共有環境と証明書発行の方針

証明書発行に関しては、以前からコミュニティの特徴にあった証明書発行手続きの模索を目的としてきた。WIDE コミュニティの特徴とは以下である。

- メンバが様々な組織から集まっている
- オフラインで会う機会は、通常 2 ヶ月に 1 回程度

この特徴は、節 2.1.1 で述べた証明書発行手続きのなかの、特に (何らかのオフライン処理が必要となる) 本人確認の難しさを示している。

WIDE メンバ間の情報共有は、メーリングリスト、WWW、(ログインして利用する) サービスホストを通じて行われている。アクセスする度にメンバ認証を必要とするのは、WWW とサービスホストである。WWW では共有パスワード登録により、サービスホストではメンバ個別の SSH の公開鍵登録により認証を実現している。オフラインで会う機会が少ない中で、共有パスワード通知時や公開鍵登録時の本人確認は厳密にはできていないが、WWW 閲覧できないと登録方法がわからないなどの暗黙的かつ現実的な方法で運用されている。

本人確認の現状に対して、証明書発行を機に厳密な本人確認をとりいれるか、現状の運用と同レベルの本人確認で証明書発行する手順を考案するかという 2 つのアプローチがある。1997 年の実験 [178] では前者のアプローチをとったため、今回は後者のアプローチをとり、運用レベルを意識して使いやすくすることに重点をおいた。

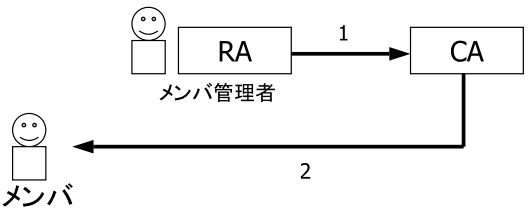

図 2.1.　moCA における CA・RA の位置づけ

2.2.2 証明書発行要求条件と CA・RA の位置づけ

現状の情報共有環境を背景として、証明書発行の要求条件を以下とした。

- メンバ管理者や PKI を応用する人にとっての管理コストをあまり増やさないようにすること
- メンバ登録作業とリンクさせ、メンバにも負担をかけないようにすること
- サービスホストへのログイン時メンバ認証レベルに近づけ、このレベルを超えないように工夫すること

このような要件を考慮すると、節 2.1.1 で述べたうちの、(a) はユーザが鍵を作成しなければならない点でメンバに負担がかかり、(b) はオフラインで鍵対を配付しなければならない点で配付コストがかかる。そこで、「(c) RA が鍵作成、かつ、オンライン配付型」をコマンド一つで実現することにした。つまり、メンバ管理者がメンバ登録を行う際に、RA としてコマンドを実行するとメンバに鍵対がオンラインで配付される、という流れである。

CA・RA の位置づけを図 2.1 に示す。

メンバに鍵対 (秘密鍵と証明書) を配付する際には、電子メールで送付するか、電子メールで WWW の URL を通知して後からメンバにアクセスしてもらうかを検討した。前者の方が、メンバにとって電子メールの環境のみで設定にとりかかれて簡単であり、CA・RA にとっても電子メール送付後は作成した鍵対を削除しやすく作業の手離れが良いため、電子メールで送付することにした。メンバは電子メールを受け取ると、WWW ブラウザに鍵対をインストールする。

2.3 moCA における実装 — ICAP と PEPOP を用いたシステムインテグレーション —

節 2.1.1 で述べたように、証明書発行の典型は「(a) ユーザが鍵作成、かつ、オンライン配付型」または「(b) RA が鍵作成、かつ、オフライン配付型」である。

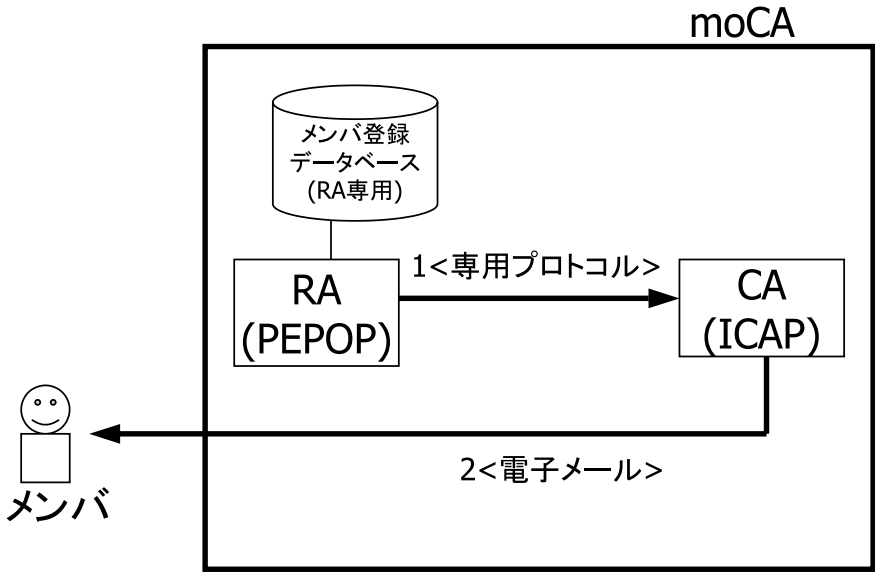

図 2.2. moCA における実装

WIDE で利用している CA パッケージ ICAP[184] は、(a) と (b) については意識しているが、「(c) RA が鍵作成、かつ、オンライン配付型」については対応していないため、拡張が必要となった。

ICAP には、専用プロトコルで連携する PEPOP(Privacy Enhanced POP)[179] というツールが別途ある。PEPOP を利用すると、RA が鍵作成をするモデルとなって今回の要件と一致することから、PEPOP に必要な拡張を加えることにした。拡張点は以下である。

- メンバ登録データベースにあらかじめ証明書に必要な情報を登録するだけで、自動的に証明書発行要求を作成できるようにすること
- 1 回のコマンド実行で n 人分の対応を行えるようにすること
- 鍵対 (秘密鍵と証明書) および CA 証明書を PKCS#12 形式に加工すること
- PKCS#12 形式のファイルを電子メールで各メンバに送付出来るようにすること

鍵対を電子メールで配付する際は、メンバのみが読めるように暗号化するが、復号化に必要なパスフレーズをどのように配付するかが最後の課題として残った。今回は、パスフレーズを別途送らずに、メンバ登録時に各メンバに割り当てられる ID を暗黙値としてパスフレーズに埋め込むこととした。つまり、節 2.1.1 で述べた (c) の本人確認のうち、以下の方法を利用したことになる。

- 既存の本人確認手段を利用して、証明書配付時に、本人だけが復号可能となるようにして配付する

実装は、PEPOP の鍵作成コマンド crtreq をベースに、openSSL の pkcs12 コマンドを組み合わせる perl スクリプトで対応した。メンバ登録データベースは WIDE のメンバ登録データベースから必要な項目を取り出して別途作成して使用した。

2.4 ユーザ対応 — アナウンス、マニュアル整備等について —

証明書発行実験を行うことについては、5 月より WIDE メンバに対して予告を行い、また、7 月の実験直前にもアナウンスを行ってユーザの理解を求めた [185]。

今回配付した PKCS#12 形式のデータは、WWW ブラウザでは鍵のバックアップを取ったり、バックアップデータをインストールしなおす際に広く使われているデータ形式である。Netscape Navigator や Internet Explorer だけでなく、大抵のブラウザに PKCS#12 形式で鍵対をインストール出来るようになった。しかしユーザインターフェイスが多少違うため、以下のブラウザにて実験用のマニュアル (あるいは tips メモ) を作成した [174]。

- Netscape Navigator 4.x, 6.0, 7.0PR1
- Internet Explorer 6.0
- w3m

● Mozilla

利用できる WWW ブラウザは 1997 年より増えたが、MacOS の Internet Explorer 5.0, 5.1 および Netscape Navigator 7.0PR1 には PKCS#12 をインストール出来なかった。

マニュアルの整備がスムーズに進んだためか、実験中のユーザからの質問等はほとんどなかった。

2.5 実験結果

今回実装したシステムを利用して、実際に 687 人のメンバに対して 1 回のコマンド実行で鍵対配付を行った。FreeBSD 4.3(CPU 600 MHz) の環境で、1,024 bit の RSA 鍵対 (PKCS#12 形式) を作成し電子メールで配付し終えるまでに 33 分 47 秒 (2027 秒)、平均すると一人あたり 2.95 秒という結果が得られた。数百人規模に配付する上では極力人手を介さずに証明書を配付できるシステムを実現できた。

今回発行した WIDE メンバ証明書のプロファイルを表 2.1、2.2 に示す。

2.6 今後の課題

秘密鍵ごと電子メールで配付することに関して安全性の課題はあるが、様々な WWW ブラウザにインストールしやすく、また、メンバの追加にも対応しやすいという点で、以前の実験に比べるとユーザおよび CA 管理者双方にフレンドリな方法へと近付いて来た。

必要なアプリケーションが整った状態で、改めて証明書発行時の本人確認方法について見直したい。また、失効管理も含めて検討したい。

第 3 章　WIDE メンバ証明書の応用

WIDE メンバ証明書の配付に伴い、moCA が発行した証明書を応用する以下の活動を行った。

- 個人情報入力/参照時の認証とアクセス制御
  合宿申込み、および、申込み内容の確認
- 認証後の、証明書の記載内容を利用した個人情報の入力省略 (ユーザの入力支援)
  – 合宿申込み内容の確認時の名前入力の省略
  – 合宿会議ごとの出席登録 (ユーザリストの作成支援)
  – WWW を利用したファイルアップロード時の作業者名自動表示
  – パーティなどのイベント申込み

本節では、これらの活動の内容を概説すると共に結果を報告する。

3.1 WIDE 合宿申込みにおける WIDE メンバ証明書の適用

WIDE 合宿は、参加申込みを行った WIDE メンバが参加する合宿である。現在、WIDE 合宿の参加申込みの受け付けは、e-side 社が用意した Web サーバと Web ページを用いて行われている。WIDE メンバは参加申込みの際に、メールアドレスや住所、請求書の送付先といった個人情報をサーバに登録する。これまで、登録された WIDE メンバの個人情報は以下の方法で保護されてきた。

1. **http + パスワード**
   HTTP を使ってアクセスし、合宿申込み (情報変更) 用のパスワードを入力する方法である。
   この方法は、https が Web ブラウザで利用できない場合や、https で使われる TCP の 443 番ポートを使った通信ができない場合に使われた。
   パスワードや個人情報が暗号化されていない状態で転送されるため、WIDE メンバへのなりすましや個人情報の漏洩の危険性が高い。
2. **https + パスワード**
   https を使ってアクセスし、合宿申込み (情報変更) 用のパスワードを入力する方法である。
   この方法は、TLS を利用するため、この通信の上での第三者がパスワードを利用して WIDE メンバになりすましたり、個人情報が漏洩することは考えにくい。しかし通信を行っているサーバが偽装されたものであると、パスワード、個人情報の両方に漏洩の危険性がある。
   この危険性をさけるため、e-side 社の Web サーバに証明書を発行して利用した。moCA によって発行されたサーバ証明書を検証するには、検証を行おうとするユーザが WIDE Root CA を信用し、その証明書を予め Web ブラウザに組み込んでおかなければならない。
3. **https + クライアント認証**
   https を使ってアクセスし、WIDE メンバ証明

表 2.1. 証明書プロファイル表

フィールド名 (フィールドの意味)	WIDE メンバ証明書
Certificate	
tbsCertificate	
version(X.509 のバージョン)	v3
serialNumber(証明書シリアル番号)	必須
signature(署名アルゴリズム)	sha1 WithRSAEncryption
issuer(証明書発行者名)	
Country (C=)	○
StateorProvince (ST=)	−
Organization (O=)	○
OrganizationalUnit (OU=)	○
CommonName (CN=)	−
DomainComponent (DC=)	−
EmailAddress (Email=)	−
validity(有効期間)	終了期日固定 (1 年)
notBefore(有効期間開始)	
UTCTime	○
GeneralizedTime	−
notAfter(有効期間終了)	
UTCTime	030630235959Z
GeneralizedTime	−
subject(証明書所有者名)	
Country (C=)	○
StateorProvince (ST=)	−
Organization (O=)	○
OrganizationalUnit (OU=)	−
CommonName (CN=)	○ WIDE 番号+” ”+氏名
DomainComponnt (DC=)	−
EmailAddress (Email=)	−
subjectPublicKeyInfo (証明書所有者の公開鍵情報)	
algorithm	rsaEncryption
subjectPublicKey	1,024bit
issuerUniqueID	−
subjectUniqueID	−

表 2.2.　証明書プロファイル表 (続き)

フィールド名 (フィールドの意味)	WIDE メンバ証明書
extensions(X.509v3 拡張フィールド)	
AuthorityKeyIdentifier (証明書発行者の鍵識別子)	○
SubjectKeyIdentifier (証明書所有者の鍵識別子)	−
KeyUsage(鍵の利用目的)	○
digitalSignature	○
nonRepudiation	○
keyEncipherment	○
dataEncipherment	○
keyAgreement	○
keyCertSign	×
cRLSign	×
encipherOnly	×
decipherOnly	×
privateKeyUsagePeriod(秘密鍵の利用期間)	−
certificatePolicies(証明書のポリシ情報)	−
PolicyMappings(ポリシ間のマッピング)	−
SubjectAltName(証明書所有者の別名)	
rfc822Name(電子メールアドレス)	○
IssuerAltName(証明書発行者の別名)	−
subjectDirectoryAttributes (証明書所有者のディレクトリ属性)	−
BasicConstraints(証明書の種別 (CA か EE か))	−
NameConstraints(名前の表現に関する制限)	−
PolicyConstraints(ポリシに関する制限)	−
ExtKeyUsageSyntax (keyUsage では表現しきれない鍵の利用目的)	−
cRLDistributionPoints (CRL 入手のためのアクセス先)	○ http
authorityInfoAccess (証明書発行者情報へのアクセス先)	−
signatureAlgorithm	sha1WithRSAEncryption
signatureValue	必須

書を使ってクライアント認証する方法である。
パスワードを利用しないため、WIDE メンバにとっての脅威は個人情報の漏洩になる。
ユーザにとって、アクセスしている https サーバのなりすましの危険性は、https+パスワードの方法と同じである。
Web サーバは、クライアント認証に使われた証明書の内容を利用することでアクセスしたユーザを識別することができる。

昨年度の 2002 年 3 月に行われた WIDE 合宿の申込みの際には、「https+パスワード」の方法が使われた。

2002 年 9 月の WIDE 合宿申込みの際には、新たに「https+クライアント認証」が使われるようになった。2002 年 3 月以前の合宿申込みの際には、WIDE メンバはアクセス後に WIDE 番号かメールアドレスを入力する必要があった。しかし「https+クライアント認証」を利用すると、証明書に格納された情報を利用して入力を軽減することができる。そこで e-side 社に協力を求め、証明書に格納された情報を用いた合宿申込み処理プログラムの作成を要請した。

具体的には、「https+クライアント認証」の方法で合宿申込みを受け付け、証明書から WIDE 番号を自動的に取り出して、以前に入力された個人情報の検索に利用するということである。ユーザは、WIDE メンバ証明書を利用して合宿申込みのページにアクセスすると、WIDE 番号やメールアドレスを入力する必要がない。

当初は意図していなかったが、登録された情報 (宿泊や食事の予約) の変更のために WIDE メンバ証明書を利用することで、専用に登録していたパスワードを利用する必要がない、というメリットが生まれた。

moCA WG では、WIDE メンバ証明書を利用したユーザの識別に注目し、2002 年 9 月の合宿では、次に述べる会議の出席簿登録の実験を行った。

3.2 会議の出席簿登録

WIDE 合宿で開かれる会議のほとんどは、WIDE メンバであればだれでも参加できる。参加者は事前に登録することなく会場に向かえばよいことになっている。そのため特定の会議に参加したメンバを知るには、その会議が開かれている間に、その場で記録を取る必要がある。会議の参加者を把握するためには、IETF などで行われているように紙に一人一人の氏名とメールアドレスを記述してもらう方法が一般的である。しかしこの方法では、書かれた文字が識別しにくい場合や、氏名を記入しないメンバがいる場合に、参加者を把握することが難しい。

そこで WIDE メンバ証明書を応用し、特定の Web ページにアクセスしてもらうことでユーザの氏名とメールアドレスのリストを作る CGI プログラムを作成した。moCA ではこのユーザのリストを出席簿登録と名づけ、出席簿登録の Web ページへのアクセス状況を調べる実験として実施した。出席簿登録の Web ページは、2002 年 9 月の WIDE 合宿期間中に利用できるようにした。

3.2.1 出席簿登録を作る CGI プログラムの処理

出席簿登録を作る CGI プログラムは、以下のような処理を行う。

1. **https のサーバが設定した環境変数を取得する**
   クライアント認証が成功すると、https のサーバは CGI プログラムを起動する際に特定の環境変数を設定する。CGI プログラムはこの環境変数を取得し、解釈することで https の認証処理の結果を得ることができる。環境変数には、クライアント証明書に格納された値が格納されている。環境変数名と値の例を以下に示す。

```
SSL_CLIENT_CN = 0481 Kimura Taiji
SSL_CLIENT_M_VERSION = 3
SSL_CLIENT_I_O = WIDE Project
SSL_CLIENT_V_START = Jun 20 01:05:22 2000 GMT
SSL_CLIENT_S_DN_EMAIL =
                    taiji-k@is.aist-nara.ac.jp
SSL_CLIENT_I_OU = members only CA
SSL_CLIENT_S_DN_CN = 0481 Kimura Taiji
SSL_CLIENT_I_DN = /C=JP/O=WIDE Project/OU=
                                members only CA
SSL_CLIENT_I_DN_OU = members only CA
SSL_CLIENT_S_DN_C = JP
SSL_CLIENT_S_DN_O = WIDE Project
SSL_CLIENT_EMAIL = taiji-k@is.aist-nara.ac.jp
SSL_CLIENT_M_SERIAL = F0
SSL_CLIENT_V_END = Jun 30 23:59:59 2003 GMT
SSL_CLIENT_I_DN_C = JP
SSL_CLIENT_C = JP
SSL_CLIENT_I_DN_O = WIDE Project
SSL_CLIENT_I_C = JP
SSL_CLIENT_O = WIDE Project
```

   出席簿登録を作る CGI プログラムは、SSL_CLIENT_CN に設定された値から氏名を取り出し、SSL_CLIENT_EMAIL に設定された値からメールアドレスを取り出す。
2. **クライアントの IP アドレスを取得する**
   クライアント (Web ブラウザ) の IP アドレスを環境変数 REMOTE_ADDR から取得する。またアクセス時刻を記録しておく。
3. **クライアントの地理的な場所を検索する**

IP アドレスと地理的な場所の表からクライアントの場所を検索する。

4. **ユーザが出席している会議を検索する**
   地理的な場所の名称と時刻からユーザが参加している会議の名称を検索する。
5. **ユーザの一覧を更新する**
   該当する BoF のファイルに、今回アクセスしたユーザの氏名とメールアドレスを一覧に加える。

出席簿登録を提供している Web サーバは、WIDE メンバ証明書を発行している WIDE 系列の CA 証明書だけを使ってクライアント証明書の検証を行っている。またこの CGI プログラムは、クライアント認証が成功しなければ上記の処理を行わない。従って、有効な WIDE メンバ証明書を利用している場合にのみ、そのユーザは出席簿登録に載ることになる。

3.2.2 利用者数

出席簿登録利用者の数を調べるため、アクセスログを取り集計を行った。出席簿登録は会議だけでなく、全体セッションなどでも利用していた。そのためすべての種類の会議を一つ一つのセッションとして集計した。またテストを行ったプログラム開発者のアクセス数を除外した。集計されたアクセス数は以下の通りである。

表 3.1.　出席簿登録のアクセス数

総アクセス数	1352
総ユーザ数	62
ユーザあたりの平均アクセス数	21
総セッション数	66
セッションあたりの平均ユーザ数	12

一ユーザあたりの平均アクセス数から、一人のユーザが何度も利用していることがわかる。WIDE メンバ証明書の総発行数が 726 であるため、総ユーザ数が 62 人であることから、全体の約 10%程度のユーザが利用していたことがわかる。

3.2.3 ファイル置き場

開発当初、出席簿登録のページは、ユーザのリストを作成するだけのプログラムであった。しかしこれではユーザにアクセスを促す効果が薄いと考えられる。そこで、HTTP のファイル送信機能を利用したファイル置き場を提供した。ほとんどの会議では、プレゼンテーションスライドなどを使って議論が行われる。BoF で使われるファイルをやりとりできる場を設けることで、会議の参加者が出席簿登録にアクセスすることを期待した。

このファイル置き場はユーザごとのアクセス制御機能をもち、ファイルを置いたユーザだけがそのファイルを削除することができる。またファイル名の横にコメントを残しておくことができる。

3.2.4 ファイル置き場のアクセス数

ファイル置き場への総アクセス数は 53 であった。また総ユーザ数は 17 であった。出席簿登録のユーザ数と比較して、ファイル置き場のユーザは少なかった。

3.2.5 出席簿登録プログラムの課題

今回の出席簿登録プログラムは、クライアントの地理的な場所を検索する際の精度があまりよくなかった。これはユーザの地理的な場所を、クライアントの IP アドレスを元に判別しているためである。無線 LAN の利用者は、意図せずに隣の基地局を利用している場合などに会議の会場とは違った場所にいると判別されてしまうことがあった。この問題は iCARS WG が 2002 年 3 月の WIDE 合宿で行なった、タグを用いた位置情報システムと連携することで改善されると考えられる。

また CGI を用いてユーザを特定するアプリケーションは、出席簿登録で行ったアクセスリストの作成やアクセス制御に限らず、様々な機能を実現することが可能だと考えられる。今後、証明書を利用したユーザの情報と時刻や IP アドレスといった情報を利用し、ユーザや状況ごとに処理内容を変更できるアプリケーションを模索することが、出席簿登録プログラムの課題である。

3.3 WIDE 外での WIDE メンバ証明書の適用：割引モデル

本実験では、WIDE 内のアプリケーションではなく、外部のアプリケーションにおける証明書利用モデルとその具体的な方式についての検討と実証実験を行った。WIDE 外部のアプリケーションとは、サービス提供主体が WIDE プロジェクトの外部であるケースを指す。本実験では、パーティへの参加登録という事例を用いて、WIDE メンバ証明書を利用し

た参加登録に対して、参加費を 1000 円割引くという設定で実験を行った。

3.3.1 場面の設定

パーティの開催者は WIDE メンバでもある個人であったが、パーティの開催自体は WIDE プロジェクトの活動とは直接関連しないものであるため、今回のサービス提供 (パーティの開催) は WIDE プロジェクトとは独立のものであると考える。一方で、パーティの主催者は WIDE メンバのパーティへの積極的な参加を希望しており、WIDE メンバに限って参加費を 1000 円割り引くという「特典」を用意することとした。

また、パーティの主催者は一般的な WWW インタフェースを用いた参加登録では、多くの入力必須な項目があり、手続きが煩雑であることを問題視していた。そこで、WIDE メンバの認証に利用するデータから、参加登録に必要な情報の一部を抽出し、自動的に入力する方式を採用し、サービス利用者 (参加登録をする人) の負担の軽減を図ることとした。

3.3.2 サービスデザイン

割引きの根拠とする WIDE メンバ認証には、moCA が配布する電子証明書である WIDE メンバ証明書を用いることとした。アプリケーションは、WWW のクライアント認証機能を利用する。WIDE メンバ証明書を利用して参加登録を行った利用者については、参加登録データベースに WIDE メンバであることが記録され、パーティ当日の受付の際に割引料金が適用されることとした。他方で、WIDE メンバであっても WIDE メンバ証明書を利用しなかった利用者に対しては、通常どおりの参加費を適用することとした。

また、参加登録時の入力負担を軽減するために、WIDE メンバ証明書に含まれている

- 氏名
- 所属 (WIDE Project)
- メールアドレス

のデータを証明書から自動的に抽出し、パーティ参加者が入力する参加登録フォームのデフォルト値とした。

3.3.3 システムデザイン

上記のサービスを実現するために、以下のようなシステムを設計・実装した。システムの基本的な構成はサーブレットの形態であり、JAVA で実装した。バックエンドの DB として PostgreSQL 7 を使用し以下のデータを配置している。

- WIDE 会員データベース
- EntryList

ここで言っている WIDEDB は実際の WIDEDB からのデータではなく、WIDE が研究者一覧なので公開しているデータから機械生成されたものである。今回の実験では、このようなデータは無くても構わないが、サービス提供者が独自のデータと連携するモデルの一環として用意している。EntryList は受け付け状況そのものを表現するテーブルである。このテーブルにサービス利用者のエントリーが存在する場合、何らかの意思表示などが行われたことを表している。まず、非証明書使用ユーザがアクセスしてきた場合、空のフォームを出力し、POST されたデータを EntryList へ上書き登録する。この動作はステートレスに動作する一般の受付システムと同じである。証明書使用ユーザがアクセスしてきた場合、EntryList から当該レコードが存在するか確認し、存在するならレコードの値をデフォルトの値としてフォームを出力する。このような動作も通常のステートフルなシステムでは一般的に行われている動作である。本システムでは独自のデータとの連携も目標としているので、EntryList にレコードが無い場合の動作も定義している。レコードが存在しない場合、証明書やバックエンドの DB からフォームのデフォルト値を収集し出力する。

このような動作によってユーザは以下のようなメリットを享受できる。

- 証明書に記載することの出来ないデータを入力する労力が軽減できる。
- 複数回入力することによる誤入力を回避できる。

3.3.4 実験結果

実験は、2002 年 9 月 13 日から 10 月 5 日にかけて行われた。この期間中、パーティの参加登録のための WWW を公開し、氏名・所属・メールアドレスを記入するフォームを利用して、参加登録を行った。期間中に参加登録をした人数は 100 名で、うち WIDE メンバ証明書を用いた登録は 15 名で、登録者全体の 15%にあたる。また、WIDE メンバであるが証明書を利用した登録を行わなかったのは 1 名で、

これは WIDE メンバである登録者のうちのおよそ 6%にあたる。

3.3.5 考察・議論

本実験では、パーティへの参加登録者の 15%が電子証明書による参加登録を行い、また、参加登録者の中で電子証明書を持つ 16 名のうち 15 名がそれを利用したという結果が得られた。これは、電子証明書を利用することによって割引きが得られるという特典が最も大きなファクターであると考えられるが、参加者全体の 15%という数字は決して小さいものではないため、効果的に利用することによって多くの会員をサービスへと誘導できる可能性があることを示している。また、逆に本実験を電子証明書利用のプロモーションという視点から考えると、イベントへの参加費割引きは電子証明書の普及に対して効果的な手法であるということができる。

WIDE メンバ証明書のような、ある組織のメンバであることを示す電子証明書を外部のサービス提供主体が利用することについては、以下のようなケースに分類できる。本実験は、この中のケース 3 という位置付けになる。

1. サービス提供主体と WIDE プロジェクトが協力関係にある場合
2. WIDE プロジェクトがサービス提供主体に対して会員のためのサービスに関して業務委託をしている場合
3. サービス提供主体と WIDE プロジェクトは直接の関係をもたないが、サービス提供主体が WIDE メンバの参加に何らかの意義を見出す場合

WIDE プロジェクトは、Global IPv6 Summit in Japan など、いくつかのイベントでの協賛・後援などを行っている。このようなイベントにおいては主催者と WIDE プロジェクトの間には 1 の関係が成り立ち、この関係をベースにした「参加割引き」のための認証手段として WIDE メンバ証明書が利用可能であると考えられる。

2 のような可能性については、WIDE プロジェクトでは想定することは難しいが、一般的な企業においては福利厚生の一環として社員向けのサービスを業務委託するというケースは十分考えられる。また、ホテル・レストラン・娯楽施設等でのクレジットカードの会員向け割引きサービスなども同じモデルであると考えられる。このようなモデルでは、会員証を示すことで特典を受けることができるが、インターネット上のサービスでこれらの特典を実現する場合には、電子証明書を利用した会員確認は有効に機能すると考えられる。

これらのサービスを実現する際の課題として、会員データベースとの連携があげられる。電子証明書を発行する団体では、独自の会員データベースを構築していると考えられる。実際、WIDE プロジェクトにおいても会員データベースが存在し、会員の氏名・所属組織・職位・研究テーマ等の情報が記録されている。外部のサービスを利用する際に、このような会員データベースとの連携が可能となれば、より簡単に利用登録を行う事ができると考えられる。具体的な例をあげると、今回の実験では、WIDE メンバ証明書から得られるデータのみを利用したため、氏名の欄には証明書に記載されているローマ字の表記が入力され、一般の利用者が入力する仮名の表記と一致しないため、参加者の管理が難しくなるという問題が生じた。このような場合に WIDE 会員データベースとの連携が可能であれば、データベースに格納された仮名表記の氏名を入力することが可能となり、より利便性が高くなると考えられる。

しかし、WIDE プロジェクトから見て外部の組織にあたるサービス提供者に対してデータベースの内容を全て公開することは、個人情報保護の観点からは望ましくない。このような場合には、会員データベース自体は自組織で保持し、外部サービス提供者との契約に基づき、特定のサーバからの問い合わせに対して、アクセスを許可している属性の値のみを返すといった、データベースへのアクセスコントロールを用いて部分的な情報の受け渡しを実現することが望ましい。このようなデータベースのアクセスコントロールの実現に際しても、電子証明書を用いたサービス認証が有用であると考えられる。

第 4 章　第 54 回 IETF における ChallengePKI 2001 の報告

4.1 ChallengePKI 2001 の報告

2001 年度の後半に、NPO 日本ネットワークセキュリティ協会 (JNSA) によって行われた相互運用実験 ChallengePKI 2001 に moCA WG のメンバが参加した。ChallengePKI 2001 の成果は、報告書としてまとめられただけでなく、第 54 回 IETF PKIX WG にて報告された。本節では、この報告の概要を述べる。

4.2 ChallengePKI 2001

moCA WG は、ICAP を用いた PKI の運用と PKI 技術の獲得の一環として、2001 年度に情報処理振興事業協会 (IPA) が公募し、JNSA が採択された「PKI 製品の相互接続実験 (JNSA Challenge PKI 2001)」に参加した。この相互接続実験は、PKI 応用製品を含む相互接続実験であり、Web サーバへの接続、S/MIME でのメール交換、IPsec での証明書を用いた暗号化/認証通信を含むものである。

4.3 実験の結果と第 54 回 IETF における報告

実験において、X.509/RFC2459 にて誤解を招く表現、認証局/PKI クライアントの実装上判断に迷う点、相互認証モデルによっては検証不能を招く表現が見つかりその点に関しての報告を第 54 回 IETF において報告し、2002 年 12 月に英語版の報告書 (サマリ) を提出した。

4.4 実験と報告に関連する文献

ChallengePKI 2001 の報告は IPA セキュリティセンター [3] で “PKI 関連相互運用性に関する調査報告”[8] および “相互運用性の現状について”[7] として公開されている。

ChallengePKI 2001 の報告書は “複数 CA 製品を使用した相互運用性に関する調査 (実証実験報告書)”[12] として公開されている。この報告書の要約を英訳したものが “Implementation Problems on PKI”[2] である。

第 5 章　WIDE ROOT CA の活動

moCA WG では、moCA, SOI CA をサブ CA とする階層型ルート CA として、WIDE ROOT CA を運用している。WIDE ROOT CA が発行するサブ CA の証明書は有効期限を 2006 年 6 月末までと長くしており、通常は年間を通じてほとんど活動場面がない。以下では、今年度の、緊急の証明書発行し直しと、新規サブ CA 証明書発行について述べる。

5.1 サブ CA 証明書発行し直し

証明書および CRL のプロファイルに関する IETF 標準 RFC2459[65] が 1999 年に発行されて以来、RFC2459 に沿った実装がブラウザ等で着実に進められてきた。また、RFC2459 の後継版である RFC3280[130] も 2002 年 4 月に発行され、相互運用性テストはますますさかんになってきている [80]。

WIDE ROOT CA では 2000 年 1 月に 6 年間有効の予定でサブ CA 証明書発行を行ったが、証明書プロファイルは必ずしも RFC2459 に沿っていなかった。2002 年 6 月の moCA での WIDE メンバ証明書プロファイルの検討時に、サブ CA 証明書プロファイルを変更しなければ、S/MIME を利用できないツールがあるとわかり、急遽サブ CA 証明書 (moCA, SOI CA) の発行し直しを実施した。具体的には、keyUsage 拡張フィールドの追加を中心に行った。

また、SOI CA の証明書に関しては、subjectKeyIdentifier 拡張フィールドの値設定ミス (但し実害なし) が発見されたため、2003 年 1 月に再度証明書の発行しなおしを実施した。

サブ証明書プロファイルを表 5.1、5.2 に示す。

5.2 新規サブ CA 証明書発行

AI3 CA が 2003 年 1 月に新規構築されることとなり、証明書発行依頼を受けて新規にサブ CA 証明書を発行した。

5.3 CA 鍵のフィンガープリント一覧

商用ルート CA では最初からブラウザにルート CA

表 5.1.　サブ CA 証明書プロファイル

フィールド名 (フィールドの意味)	2002 年 6 月まで	2002 年 7 月以降
Certificate		
tbsCertificate		
version(X.509 のバージョン)	v3	v3
serialNumber(証明書シリアル番号)	必須	必須
signature(署名アルゴリズム)	SHA-1 WithRSAEncryption	SHA-1 WithRSAEncryption
issuer(証明書発行者名)		
Country (C=)	○	○
StateorProvince (ST=)	−	−
Organization (O=)	○	○
OrganizationalUnit (OU=)	−	−
CommonName (CN=)	○	○
DomainComponent (DC=)	−	−
EmailAddress (Email=)	−	−
validity(有効期間)	終了期日固定	終了期日固定
notBefore(有効期間開始)		
UTCTime	○	○
GeneralizedTime	−	−
notAfter(有効期間終了)		
UTCTime	060630235959Z	060630235959Z
GeneralizedTime	−	−
subject(証明書所有者名)		
Country (C=)	○	○
StateorProvince (ST=)	−	−
Organization (O=)	○ (optional)	○ (optional)
OrganizationalUnit (OU=)	○ (optional)	○ (optional)
CommonName (CN=)	○ (optional)	○ (optional)
DomainComponnt (DC=)	−	−
EmailAddress (Email=)	−	−
subjectPublicKeyInfo (証明書所有者の公開鍵情報)		
algorithm	rsaEncryption	rsaEncryption
subjectPublicKey	2,048bit	2,048bit
issuerUniqueID	−	−
subjectUniqueID	−	−

表 5.2. サブ CA 証明書プロファイル (続き)

フィールド名 (フィールドの意味)	2002 年 6 月まで	2002 年 7 月以降
extensions(X.509v3 拡張フィールド)		
AuthorityKeyIdentifier (証明書発行者の鍵識別子)	−	−
SubjectKeyIdentifier (証明書所有者の鍵識別子)	○ (moCA のみ)	○
KeyUsage(鍵の利用目的)	−	○
digitalSignature		○
nonRepudiation		○
keyEncipherment		○
dataEncipherment		○
keyAgreement		○
keyCertSign		○
cRLSign		○
encipherOnly		×
decipherOnly		×
privateKeyUsagePeriod (秘密鍵の利用期間)	−	−
certificatePolicies(証明書のポリシ情報)	−	−
PolicyMappings(ポリシ間のマッピング)	−	−
SubjectAltName(証明書所有者の別名)	−	−
IssuerAltName(証明書発行者の別名)	−	−
subjectDirectoryAttributes (証明書所有者のディレクトリ属性)	−	−
BasicConstraints (証明書の種別 (CA か EE か))	○ critical,CA	○ critical,CA
NameConstraints (名前の表現に関する制限)	−	−
PolicyConstraints(ポリシに関する制限)	−	−
ExtKeyUsageSyntax (keyUsage では表現しきれない 鍵の利用目的)	−	−
cRLDistributionPoints (CRL 入手のためのアクセス先)	−	○ http
authorityInfoAccess (証明書発行者情報へのアクセス先)	−	−
signatureAlgorithm	sha1With RSAEncryption	sha1With RSAEncryption
signatureValue	必須	必須

証明書が登録されているが、実験用ルート CA では、ユーザがインストールしなければならない。CA 鍵のフィンガープリントは、CA 証明書をインストールする際に、CA を信用するかどうかを判断するために必要な確認情報として使われる。フィンガープリントの確認は、ブラウザの CA 証明書表示機能、または、CA 証明書を PEM 形式で保存した後 openSSL コマンド実行によりできる。

openSSL の場合の確認方法は以下である。

```
% openssl x509 -fingerprint -md5 -noout
                              -in cacert.pem
% openssl x509 -fingerprint -sha1 -noout
                              -in cacert.pem
```

以下に、2003 年 1 月現在の各 CA 鍵のフィンガープリントを示す。

- WIDE ROOT CA
  - MD5 フィンガープリント
    `2B:68:BD:1B:26:28:2A:AC:CF:F3:45:90:1D:6C:2A:9C`
  - SHA-1 フィンガープリント
    `3560 185D 83DC CBB7 0EBB 45AD 1E9B F529 A816 0562`
- moCA
  - MD5 フィンガープリント
    `17:FD:D2:8A:C2:36:5D:0E:0B:A7:69:BC:9D:7F:E6:97`
  - SHA-1 フィンガープリント
    `487E 16E1 746E 5C16 8A7D C55D DE80 37E8 9241 7FA3`
- SOI CA
  - 2003 年 1 月現在
    - MD5 フィンガープリント
      `8E:D0:BD:BD:DD:FE:A6:34:23:12:99:95:75:1F:69:2F`
    - SHA-1 フィンガープリント
      `29A8 AEEA C1B7 78B0 28CF 3F1C 1D98 7ED8 6203 DCA8`
  - 2003 年 4 月以降 (予定)
    - MD5 フィンガープリント
      `7B:23:02:D1:76:37:44:81:76:35:DA:8A:51:BF:B5:48`
    - SHA-1 フィンガープリント
      `0A92 34A8 B589 C835 6101 3151 CBC6 4F18 1ACE 6D4D`
- AI3 CA
  - MD5 フィンガープリント
    `19:89:C9:CF:D5:E1:8F:E1:65:51:92:72:A2:49:96:0F`
  - SHA-1 フィンガープリント
    `0AE8 76F5 7240 BA67 99B9 A200 C94C 1650 FBB5 29F0`

第 6 章　moCA WG の課題

今年度はプライベート CA の追求として、証明書配付方法の工夫だけでなく、証明書の利用を促進するための、アプリケーション模索に注力した。証明書を配付さえすれば使い方は自然と出てくるという考え方を改め、moCA WG 活動の中で自ら利用場面を考え、提案し、実現する活動へと方向転換を図った。

PKI による認証のよさを、厳密個人認証とのみ結びつけるアプリケーションで示すことは難しく、認証した後に、証明書の記載内容を利用して次のアクションに繋げるアプリケーションをいくつか開発した。証明書を日常の活動に使えるようにするには、組織の活動に沿った、便利に使えるアプリケーションをもっと用意する必要があると考えている。来年度も、PKI アプリケーションの開発に注力し、多くの人が使える WWW の環境を中心に、CGI を含めたアプリケーションの開発を進めたい。また、WG 内での活動にとどまらず、今後も、他の WG との連携を積極的に図っていきたい。

アプリケーションをできるだけ多くの人が使えるようにするためには、PKI 要素技術に関する製品実装状況の把握にも努める必要がある。今年度はサブ CA 証明書発行し直しを行ったが、その決断には、JNSA 主催の相互接続実験参加を通じて得られたノウハウによるところが大きかった。来年度も WIDE 以外での活動や PKI 技術者とのリンクを生かして、最善の環境でアプリケーションを展開できるようにしたい。