IC カードを用いた共有端末認証
― IC カードを利用してユーザごとの作業環境を構築する ―

葛　生　和　人

Ⅰ．はじめに

クレジットカード，電子マネー，鉄道の自動改札カードなど，その便利さから近年急速に広まったIC カードですが，ネットワークが普及して情報セキュリティへのニーズが高まっていることを考えると，コンピュータユーザにとってもその認証機能の活用は大きな魅力です。実際，行政が進める公的個人認証サービス（電子申請，電子入札など），多くの企業で採用されつつあるIDや入退室管理，PC 端末へのログオン，アプリケーションへのアクセス認証など，ネットワーク上で個人を特定するためのIC カードの利用法は今後ますます重要となるでしょう。

ところが，このように今では多くの人に馴染み深いIC カードですが，カードアプリ開発に関わる環境の整備，システムの標準化は十分には進んでいません。もちろん，カードそのものの仕様に関するスタンダード自体は規定されてはいますが，実際にカードアプリが内部でどのように構築されていて，どのような動きをしているのかは，カードアプリケーションの開発ベンダー以外はなかなかわからないというのが現状です。それは，セキュリティの観点からは望ましいことかもしれませんが，カードを利用するユーザの立場からすると，ユーザが独自に新たな利用法を探る手立て，アプリケーションを開発する道を閉ざしてしまうことになり，IC カードのさらなる普及を阻害することとなりかねません。

そのような中で，汎用性のあるカードアプリ開発をめざせる 1 つの解決策として Java Card™ Technology [1] を利用したカードアプリ開発は注目に値するでしょう。実際，この Java Card™ Technology の開発環境を利用して，PKI と連携した Windows スマートカードログオン用の IC カードアプリを実装する方法については，当センターニュース [2] で解説しています。

今回は，そこで開発した Windows スマートカードログオン用 Java Card アプリに対して，さらに新たな利用法を探り，利便性という付加価値を付け加えた形のアプリケーション事例を紹介します。それは，IC カード認証によって実現される共有端末利用法の 1 形態で，ユーザごとの作業環境を構築できるようにするためのシステム構築の例です。ここでは，その全体的な考え方と実装方法について解説します。

Ⅱ．共有端末の利用法

複数のユーザが利用可能な共有端末は，図書館等の公共施設に設置されている端末をその代表例として挙げることができます。そのような共有端末では，通常，セキュリティの観点からデス

クトップ画面がユーザログオンの状態でオープンになっており，マウスや画面の操作制限が端末ごとに厳重に管理されています。また，そこでは利用可能なアプリケーションも制限され（蔵書検索用ソフトのみに使用が制限されるなど），ユーザが自由にログオフすることもできません。このようなセキュリティ上のシステム管理は，部外者も含めて広範な利用者を想定した公共施設の共有端末であれば当然のことでしょう。

しかし，関係者の入退出がある程度制限された中小規模の施設では，もう少し自由度を持った利用方法，セキュリティポリシの設定も考えられます。例えば，ログオンログオフはユーザがその都度行える，あるいは，施設が提供する検索システムだけではなく，一般のインターネット検索エンジンや，文献のダウンロードシステムが利用できるなど，ある程度の利便性を追求した使用を許可する運用形態もあるかもしれません。

当然その場合は，セキュリティの観点からログオン時のユーザ認証に配慮する必要が出てきます。そこで注目されるのが，IC カードを用いたログオン認証です。つまり，IC カード認証を利用することにより，共有端末利用者の素性を保証し，その代わりに端末利用者にある程度自由度のある作業環境を提供する，そのような共有端末利用形態が可能となってきます。

そこで上に述べたような共有端末の利用形態を仕様としてまとめてみると以下のようになります。

①端末利用者は IC カードの所有者に限定される。

②端末利用者の権限はゲストユーザ権限に限定される。

③どの共有端末でもログオン時にはユーザ独自の作業環境が構築される。

④管理者はユーザの作業ログ，作業環境などいつでもチェックできる。

実は，共有端末上でユーザごとの作業環境をセットアップできるような利用形態は，Windows 系サーバを通して，クライアント PC 群で構成されるドメインを構築することにより実現することができます。そこでは，移動ユーザプロファイルという概念 [4] を利用することにより，個別ユーザの作業環境がドメインコントロールサーバ側に保存され，登録したユーザアカウントに対応した環境がどの共有端末においても再現可能になります。さらに，Windows が提供する Active Directory 連携のスマートカードログオン機能と組み合わせることにより，先に示した共有端末の利用形態は実現されるでしょう。さらに，同様のアプローチは，Windows 系サーバ以外でも Samba のドメインコントローラ機能 [5] を代替サーバとして活用すれば，Linux/UNIX 系マシンを通しての利用も可能です。図 1 は，共有端末用の移動ユーザプロファイルがドメインコントロールサーバで管理されている状態を示したものです。

しかし，LDAP などのディレクトリサーバを用いてユーザ情報を独自に管理しているようなシステムがすでに構築されているような場合，上に示したような共有端末利用法を実現するためには，新たにドメインコントロールサーバを導入しなければなりません。加えて，クライアントアクセスライセンスの獲得，個別利用アカウントの設定などディレクトリ情報の 2 重管理的な作業が発生し，経済性の面でも運用面でも多大な労力を要することとなってしまいます。

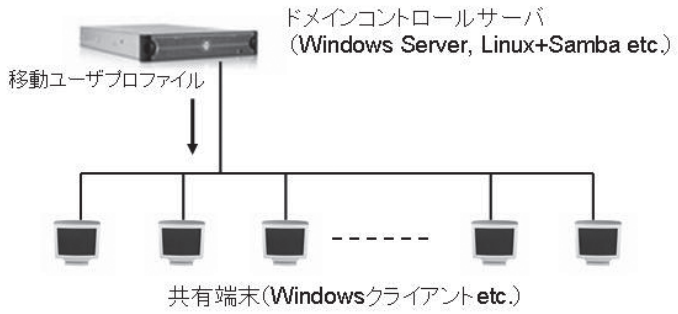

図１　ドメイン構成による共有端末

Ⅲ．作業環境とユーザプロファイル

１．ユーザプロファイルの種類

ここで，前節で出てきた作業環境とユーザプロファイルについて簡単に説明します。

Windows 系システムにおいてユーザの作業環境はユーザプロファイルにより管理されますが，このユーザプロファイルは，その機能によって，ローカルユーザプロファイル，移動ユーザプロファイル，固定ユーザプロファイルの３種類に分類されます [4]。これらプロファイルのうち，ローカルユーザプロファイルはユーザのスタンドアロンマシン上に格納されますが，移動ユーザプロファイルや固定ユーザプロファイルはドメイン管理サーバ上に格納されるため，クライアントマシンが異なったとしてもそこでユーザごとに作業環境が再構築されることとなります。なお，図

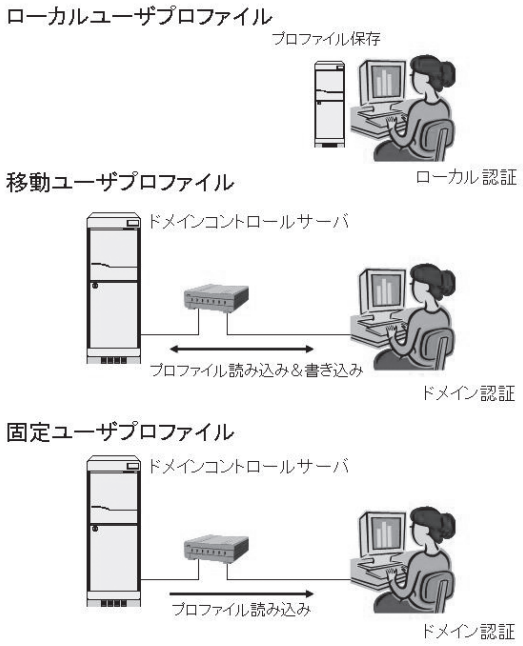

図２　ユーザプロファイルの分類

2 に示すように，移動ユーザプロファイルは登録ユーザごとにプロファイルの書き換えが可能で個々の作業環境が再構築されますが，固定ユーザプロファイルはサーバ管理者が 1 つのプロファイルを管理し，複数のユーザが同じ作業環境を共有するものです。

これらユーザプロファイルの内，ドメインサーバで管理される移動ユーザプロファイルを利用すれば，共有端末の利用形態を満足させるシステムを構築することは可能です。しかし，前節で示したように，LDAP などのディレクトリサーバを用いてユーザ情報を管理運用しているようなシステムがすでに存在している場合，新たに別のドメイン管理サーバを導入することから生ずる経済面，管理運用面での負担が問題となります。

2．ドメイン構成を利用しない移動ユーザプロファイル

そこで，ドメインを新たに構築することなく移動ユーザプロファイルの利用できる環境を考えてみましょう。

Windows システムにおいてユーザ個々の作業環境は，ログオン認証時にユーザプロファイルデータ（NTUSER.DAT）に含まれる作業環境パラメータをレジストリハイブ（HKEY_CURRERNT_USER）にロードすることにより実現されます。したがって，プロファイルデータをどこかのストレージに格納しておき，ログオン認証前にそのデータを読み込んで，システムユーザ権限でレジストリハイブへのロードすることができれば，例え共通のゲストユーザログオンであったとしても，個別ユーザとしての作業環境がローカル端末上に再構築されるようになります。

この考え方は，センターニュース [2] で解説している Windows スマートカードログオンシステムのミドルウェアを拡張する形で実現することができます。なぜなら，この Windows スマートカードログオンシステムは，winlogon.exe の認証用拡張モジュールである GINA（Graphical Identification aNd Authentication）[6] を使用して構築されており，そのためログオン認証前後のプロセスをユーザが独自に拡張することが可能だからです。つまり，ログオン認証前にユーザプロファイルをネットワーク上の別のストレージから取得してローカルファイルシステム上に展開できさえすれば，レジストリハイブへのロードプロセスはシステムが自動で行ってくれ，個別ユーザの作業環境が再構築されるようになります。そこでは，ログオン認証そのものは，ドメイン認証である必要はなく，ローカルマシンのゲストユーザとして行われるだけで十分です。

また，このシステムはもともと IC カード認証を行っているため，各ユーザの IC カードに格納されたユーザ ID を通して，読み込むべきユーザプロファイルとカード所有者の紐付けが可能となり，さらに，PKI と連携した証明書検証を行っているため，高いセキュリティが確保されることになります。図 3 は，ゲストユーザ用端末と PKI 認証で利用される LDAP サーバ，ユーザプロファイル格納用のストレージ，さらに，それぞれの間でのデータのやりとりを示したものです。

なお，図 3 から明らかなように，すでに LDAP などのディレクトリサーバを用いてディレクトリ情報を管理，運用しているような環境では，このシステムはそれらサーバ，情報資源を流用することができます。すなわち，Windows 系ドメインシステムの新規導入によって生ずる別系

統のディレクトリ情報管理，及びそれに伴うシステム運用の煩雑さを避けることができるという点で大きなメリットとなります。

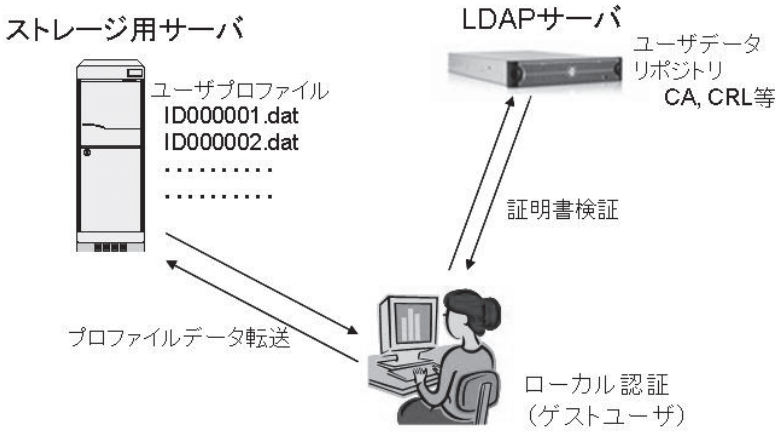

図３　ID で識別可能なユーザプロファイルの格納

Ⅳ．IC カード認証と移動ユーザプロファイルの連携

1．IC カード認証ミドルウェア

前節で示したように，本システムではログオン認証前に移動ユーザプロファイルをストレージから呼び出して，システムにセットアップする手順を組み込む必要があります。このプロセスは，センターニュース [2] で解説しているスマートカードログオン用ミドルウェアの拡張により実現できます。

なお，ここで使われる IC カード認証ミドルウェアの実行プロセスについて簡単に説明しておきます。このミドルウェアは，winlogon.exe の認証用拡張モジュール（GINA）をベースに，OS とユーザ間のインタフェース機能拡張の形で構築されています。実際の GINA の実行シーケンスを図 4 に，機能として拡張したスマートカードログオンプロセスに関わる IC カードアクセスルーチンや LDAP サーバ通信ルーチンのプロトコルを図 5 に示します。

スマートカードログオンプロトコルに示される手続きは，実際には GINA の実行シーケンス（図

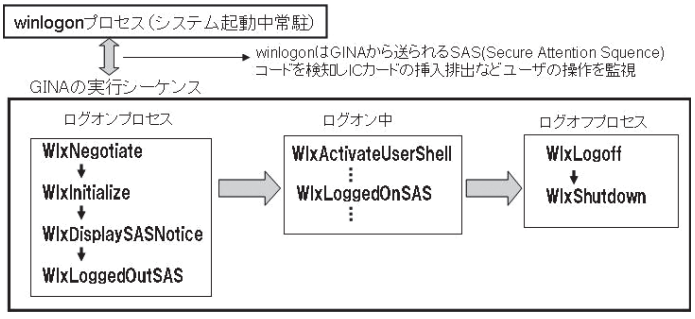

図４　GINA の実行シーケンス

4）の中では，WlxLoggedOutSAS で処理されます。なお，この時点でストレージサーバに格納された個別ユーザプロファイルは IC カード内のユーザ ID とファイル名を通して紐付けされているものとします。したがって，ログオン時のユーザプロファイル取得と，IC カード認証を通したユーザ ID の取得はここで連携されることとなります。図 6 は，上で述べた各情報の取得タイミングを示したものです。

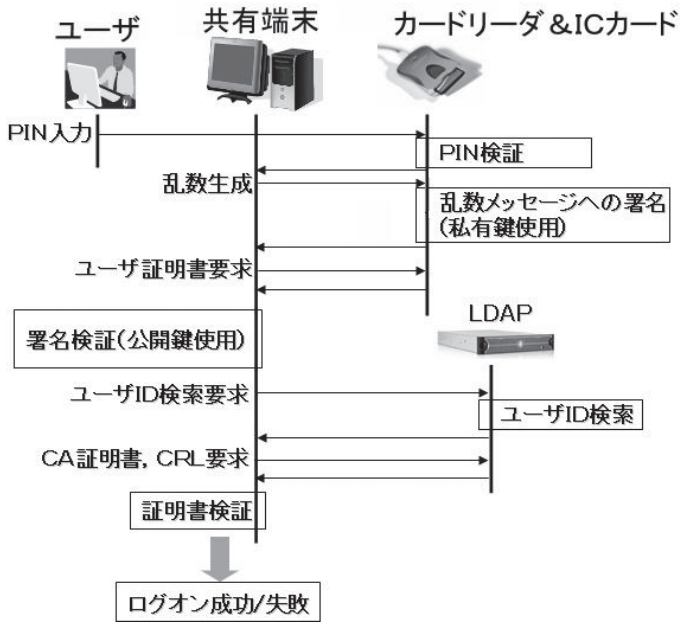

図 5　スマートカードログオンプロトコル

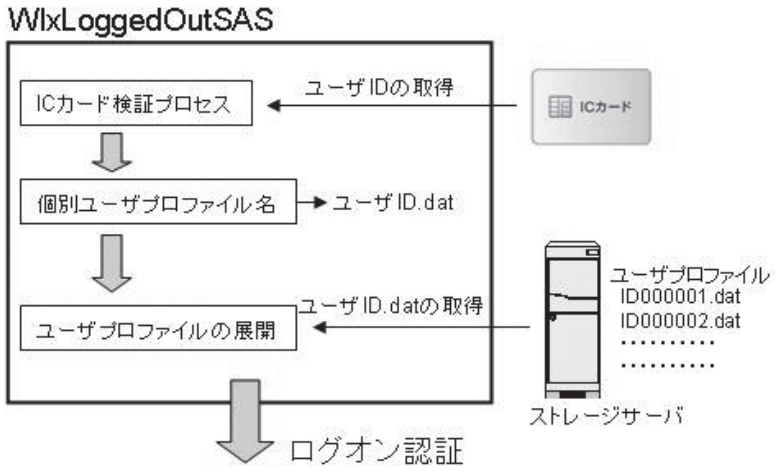

図 6　ID 番号から個別プロファイル名指定

2．移動ユーザプロファイルのストレージへの格納

作業環境の再構築に必要なユーザプロファイルは，どこの共有端末からも参照可能なストレージに格納しておく必要があります。この場合のデータストレージ用サーバは，Windows，Linux いずれのシステムでも問題ありませんが，個人情報がネットワーク上を流れるため ssh のような暗号化プロトコルの利用できるシステムであることが望ましいでしょう。ここでは Linux 上の ssh サーバを利用します。

なお，ssh サーバへのアクセスは，通常クライアントであるユーザの権限で行われますが，本システムではミドルウェアからシステム権限で直接アクセスされるため，一般ユーザ権限でストレージサーバにアクセスされることはありません。また，パスワード情報が開示されることもないためセキュリティが確保されます。

一方，実際に格納されるユーザプロファイル情報は Windows ではシステムドライブの Documents and Settings 中のユーザ名で表されるフォルダに納められています。これらは，図 7 に示したようなフォルダ群及びファイルで構成されていますが，データ格納時の形式がファイル構成に影響することはないため，ここではデータ格納前に zip 形式でパッケージングし転送する形式をとります。

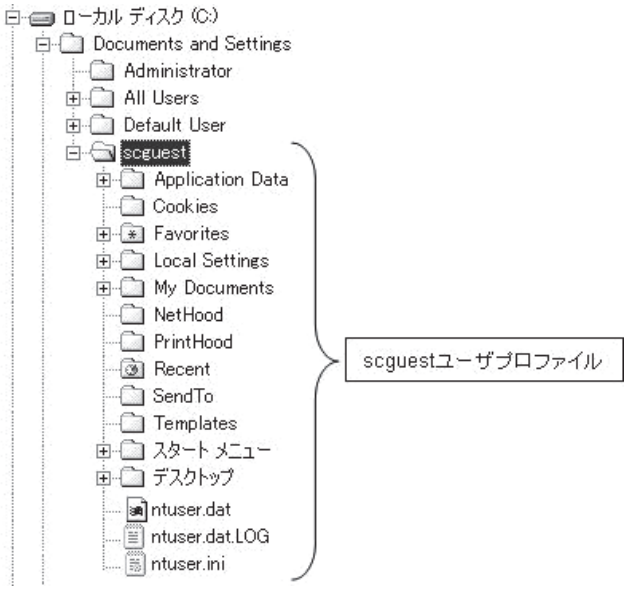

図 7　ユーザプロファイル構成

3．アクセス権限とユーザプロファイル

Windows のファイルシステムである NTFS は，それぞれのファイルやフォルダに対してアクセス権限が細かく設定されます。共有端末のゲストユーザとして登録されるユーザプロファイルに対しても同様で，アクセス権限は，通常のそのユーザ自身によるプロファイルの読み出し，変更が許可されています。しかし，共有端末のように異なる端末からログオンした場合，

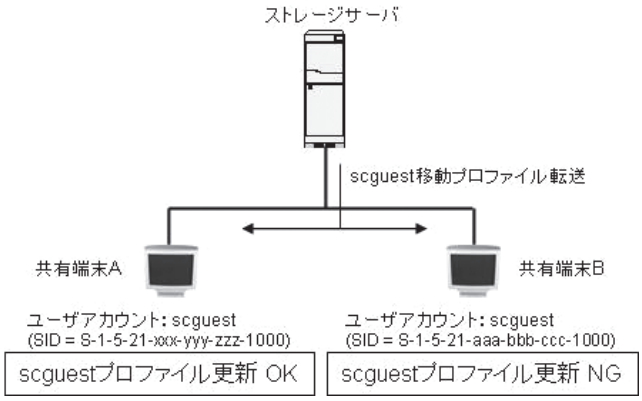

図８　異なる SID でのプロファイルへのアクセス

Windows では，同じゲストユーザであってもシステム上異なる SID（セキュリティ識別子）[7] が割り当てられます。したがって，図 8 に示すように共有端末上に同じユーザアカウントでユーザプロファイルを展開しながら，その変更権限が失われてしまう可能性があります。

ドメイン構成下での移動ユーザプロファイルに対しては，ユーザのアクセス権限はドメインにより管理されますが，今回のシステムでは，上のような状況を回避するためにあらかじめすべてのユーザプロファイルに対する読み出し，変更権限を有効にしておく（Everyone: フルコントロール）必要があります。これは，NTFS を利用したシステムでは，Windows のファイルプロパティ属性メニュー（図 9）により設定可能です。

なお，上記のようなアクセス権限の設定は，ローカルマシン上にプロファイルが残されるため，

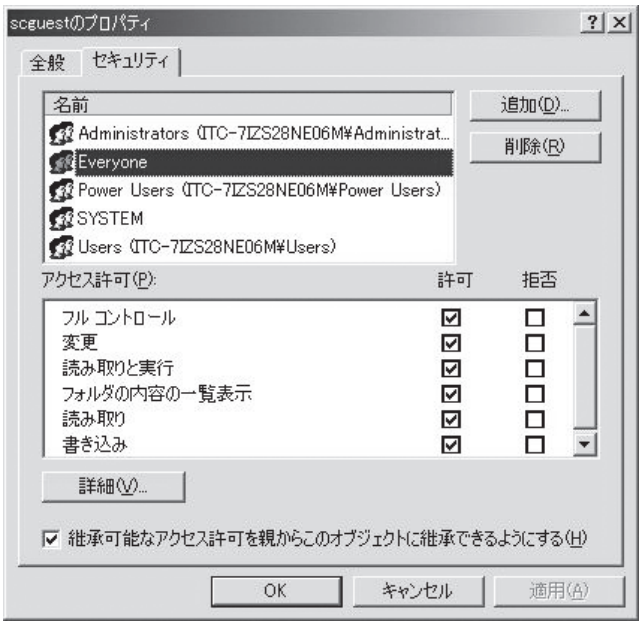

図９　プロファイルへのアクセス権限設定

ゲストユーザや管理者以外のユーザが共有するような通常の端末では，セキュリティや個人情報の保護の観点から好ましいものではありませんが，本システムにおいては，ログオフ時にはプロファイルはストレージサーバ上に転送され，ローカルマシン上に残ることはありません。また，ログオン中のプロファイルデータ及びそれを含むフォルダに関しては，ファイルシステムとしてネットワーク共有の設定を行わず，リモートデスクトップ（Windows XP 使用時）の接続を許可しないことにより，他のマシンからのアクセスも不可能となります。すなわち，ストレージサーバ上のプロファイルデータはそのサーバ管理者または ID 番号と紐付けされたゲストユーザが端末へログオンするときのみアクセス可能となり，一般ユーザからは保護されることになります。なお，図 10 はログオンプロセス時にプロファイルがレジストリに登録され，ログオフプロセス時にプロファイルデータが端末から削除されることを示したものです。

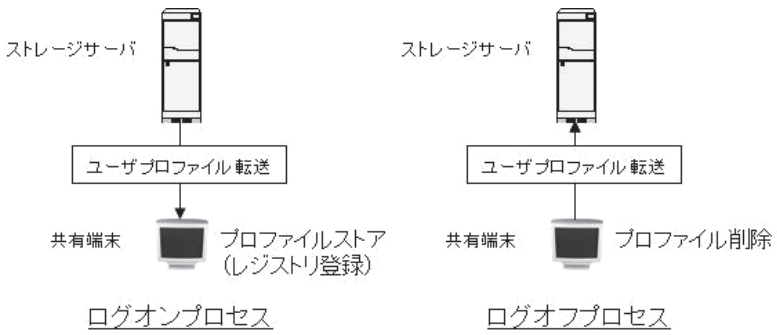

図 10　ストレージ上プロファイルの保護

4．移動ユーザプロファイル通信プロセス

前節で述べたように，ユーザ ID で識別されるゲストユーザプロファイルは，ログオフ時に

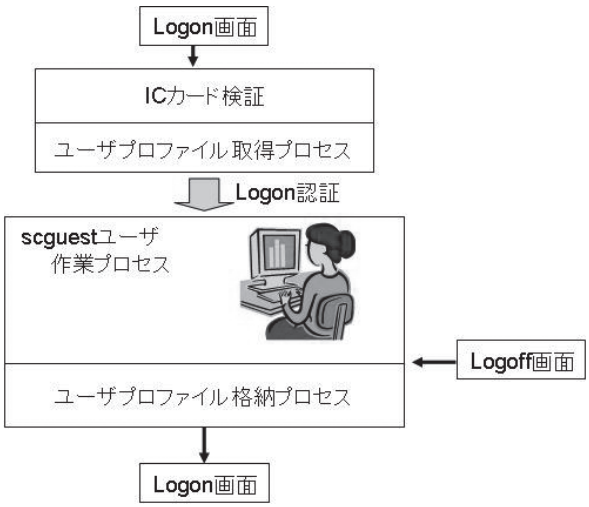

図 11　プロファイルの取得，格納プロセス

zip コマンドによりまとめられ，ssh 通信プロトコルを通してストレージに送信されます。そして，ユーザのログオン時には，逆にストレージよりユーザ ID に相当するプロファイルデータが取得され，ローカルシステムに展開されます。これらのプロセスは，実際には GINA の実行シーケンス中で外部プロセスとして実行されます。図 11 はプロファイルの取得，格納のための外部プロセスとユーザのログオン，ログオフのタイミングを示したものです。

ここで，ログオン直前のプロファイル取得に関しては，プロファイル取得以前に IC カード認証による ID 番号の抽出が行われている必要があるため，IC カード検証ルーチンの後に呼ばれ，該当するプロファイルデータを獲得，システム上に外部プロセスを通じて展開することとなりま

```
ZeroMemory(&si_prof, sizeof(si_prof));
ZeroMemory(&pi_prof, sizeof(pi_prof));
si_prof.cb = sizeof(si_prof);

CreateProcessA(NULL, "pscp.exe
                        ストレージ上プロファイルデータ
                        "c:¥Documents and  Settings¥scguest.zip"",
                        NULL, NULL, FALSE, 0, NULL, NULL, &si_prof, &pi_prof);

WaitForSingleObject(pi_prof.hProcess, INFINITE);
CloseHandle(pi_prof.hThread);
CloseHandle(pi_prof.hProcess);

ZeroMemory(&si_prof, sizeof(si_prof));
ZeroMemory(&pi_prof, sizeof(pi_prof));
si_prof.cb = sizeof(si_prof);

CreateProcessA(NULL, " unzip.exe
                         "c:¥Documents and  Settings¥scguest.zip"  -d c:¥"
                        NULL, NULL, FALSE, 0, NULL, NULL, &si_prof, &pi_prof);

WaitForSingleObject(pi_prof.hProcess, INFINITE);
CloseHandle(pi_prof.hThread);
CloseHandle(pi_prof.hProcess);
```

リスト１　プロファイル取得の外部プロセス呼び出し

```
ZeroMemory(&si_prof, sizeof(si_prof));
ZeroMemory(&pi_prof, sizeof(pi_prof));
si_prof.cb = sizeof(si_prof);

CreateProcessA(NULL, " zip.exe
                        "c:¥Documents and Settings¥scguest.zip"
                        "c:¥Documents and Settings¥scguest"",
                        NULL, NULL, FALSE, 0, NULL, NULL, &si_prof, &pi_prof);

WaitForSingleObject(pi_prof.hProcess, INFINITE);
CloseHandle(pi_prof.hThread);
CloseHandle(pi_prof.hProcess);

ZeroMemory(&si_prof, sizeof(si_prof));
ZeroMemory(&pi_prof, sizeof(pi_prof));
si_prof.cb = sizeof(si_prof);

CreateProcessA(NULL, "pscp.exe
                         "c:¥Documents and  Settings¥scguest.zip"
                        ストレージ上プロファイルデータ",
                        NULL, NULL, FALSE, 0, NULL, NULL, &si_prof, &pi_prof);

WaitForSingleObject(pi_prof.hProcess, INFINITE);
CloseHandle(pi_prof.hThread);
CloseHandle(pi_prof.hProcess);
```

リスト２　プロファイル格納の外部プロセス呼び出し

図 12　公開鍵情報のレジストリへの格納

す。なお，実際の外部プロセス呼び出しは WIN32 API に含まれる CreateProcess を用いており，そのルーチンをリスト 1 に示します。また，ログオフ直前のプロファイル格納に関しても同様で，リスト 2 のような外部プロセス呼び出しを通じて，ストレージサーバ上に格納されます。

一方，本システムではプロファイルデータの通信プロセスとして ssh サーバを利用していますが，通常，Windows システムにおいてクライアント側に格納される ssh 通信用の公開鍵情報は，カレントユーザのレジストリにロードされます。したがって，本システムのように，ログオン前にシステムユーザが外部プロセスとして ssh 通信を行う場合には，システムユーザとして使用するレジストリにあらかじめ公開鍵情報を格納しておく必要があります。実際には，Windows システムユーザがデフォルトとして参照するレジストリキー HKEY_USERS¥DEFAULT¥Software に対して図 12 のように公開鍵情報を設定します。

V．システム仕様と共有端末への実装

本システムを検証するに当たって利用した各ソフトウェア，ハードウェアの仕様とシステムの実装手順について説明します。

1．IC カード，カードリーダ及びカードアプリ

IC カードは，接触，非接触のいずれにも対応したデュアル・インタフェース型 1MB メモリを有するものでネイティブプラットフォーム上に Java Card VM を組み込んでいます。カード内でのセキュリティ API は RSA, DES, T-DES の複数の暗号処理が可能です。詳細の仕様は，表 1 に示すとおりです。また，カードリーダは，想定した 2 台の共有端末のそれぞれに対して接触型，非接触型のものを用意しました。（表 2）

ここで，Java Card VM 上に搭載する認証用アプリケーションは，Java Card Technology を使って作成したもので [2][3]，カード内には PIN コード，ユーザ証明書，ユーザの私有鍵が格納されています。ユーザ証明書は，X.509 標準規格に従い，ASN.1, DER フォーマットでエンコードされたもの，私有鍵に関しては 1024 ビット長の RSA 暗号鍵を使用しています。

なお，本実装は Java Card VM 上の Java Card アプリとそれに対応するミドルウェアで構成さ

れていますが，IC カード機能として必要な要件は，証明書の格納と PKI 認証に必要な暗号処理機能を満たすということで，Java Card™ 仕様に限定されるものではありません。

表 1　IC カード仕様

タイプ	Java Card VM	
	接触型	非接触型
準拠規格	ISO/IEC7816	ISO/IEC 14442 Type B
通信プロトコル	T = 0, 1	ISO/IEC14443-4
通信速度（kbps）MAX	19.2	424.0
メモリ	1M バイト（フラッシュメモリ）	
CPU	16 ビット	
セキュリティ	RSA, DES, T-DES 演算対応	

表 2　カードリーダ仕様

タイプ	接触型	非接触型
品名（品番）	GemPC TWIN	PD2992P
準拠規格	ISO/IEC7816	ISO/IEC　14442 Type B
インタフェース	USB 2.0	USB 1.1

2．ミドルウェア構成及び共有端末への実装

ここで採用しているスマートカードログオン用ミドルウェアは，認証，暗号化に関わる API として Windows の CriptoAPI を使用しています。また，その他 IC カードリーダとの通信，LDAP クライアントとしての通信プロセスに関しても WIN32 API を利用しています。開発環境は，VC++ ver.6 及び Platform SDK を使用しました。

一方，ここでの検証は仮想共有端末として 2 台のデスクトップマシン（ThinkCentreA52T 及び DELL Precision 650 で OS は Windows 2000 Professional SP4）を利用しています。それぞれ

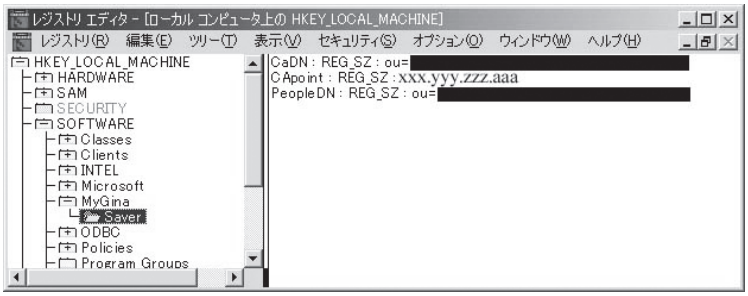

図 13　GINA レジストリ登録パラメータ

のマシンに，今回作成したスマートカードログオン用ミドルウェアをセットアップし，winlogon プロセスから参照されるようにリポジトリ情報をレジストリ HKEY_LOCAL_MACHINE ¥SOFTWARE¥MyGina に登録します。（図 13）

3. 認証局，証明書

本システムを検証するに当たり，プライベート認証局，及び CA 証明書や失効リスト（CRL）を格納するためのリポジトリサーバを 1 台準備します。サーバ用ハード及び OS の仕様は，DELL Power Edge 2850 3.8GHz Xeon プロセッサ , Cent OS5，認証局は NAREGI-CA[8] をセットアップして，CA 証明書，ユーザ証明書，CRL の発行を行っています。

また，各証明書，CRL のリポジトリとして同サーバに OpenLDAP2.3[9] を実装し，CA 証明書，CRL とともに仮想ユーザ情報も格納しています。なお, 図 14 はリポジトリに格納された仮想ユーザ情報を LDAP 用ブラウザにより表示したものです。

4. プロファイル格納用ストレージ

ユーザプロファイル格納用ストレージは，本システムにおいては前述の認証局，リポジトリと共用のものとして設定しています。したがって, 実際には同サーバ上にプロファイル格納用にディレクトリを確保し ssh プロトコルを通してデータ通信を行うものとしました。なお，これらの設定は，システム検証のために簡易的に行っているものです。

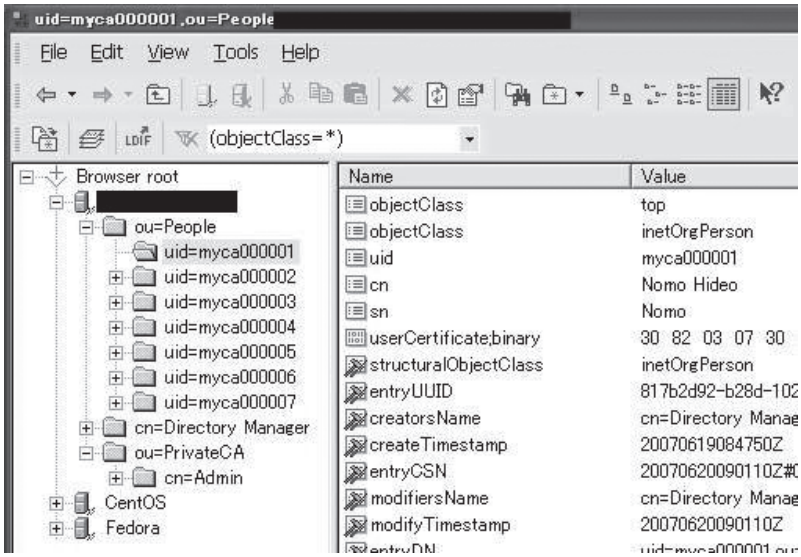

図 14　LDAP ブラウザによるリポジトリ情報の確認

Ⅵ. 実証実験

以上のように構築したシステムに対して，実際のネットワーク環境を通した実証実験を試み，

その使い勝手や機能について検証します。実験は，異なる 2 ユーザ（2 枚の IC カードに対して個別のユーザ証明書，私有鍵を格納），異なる仮想共有端末 2 台を準備して，

①異なるユーザが同一の共有端末からログオンした場合のそれぞれの作業環境の構築状況

②同一ユーザが異なる共有端末からログオンした場合の作業環境の保持状況

の 2 点に着目し検証を行います。

また，本システムではローカルマシンへの通常ログオンと比較して，プロファイルの取得，格納プロセス時間分だけログオン時間がかかることが予想されます。そのことに対するアクセス時間の比較も同時に行い使用上の改善点の抽出も試みます。

1．異なるユーザの同一共有端末作業環境

図 15 は共有端末起動後の IC カードの挿入を促す画面から PIN コード入力，さらに 2 枚の異なる IC カード挿入に対応したそれぞれのデスクトップの画面を示しています。作業環境の差異を明らかにするためにデスクトップの背景イメージとして異なる画像を用いており，ユーザに対応した環境が再現されていることが視認されますが，スタートアップメニュー，ブラウザのパラメータ，履歴，ブックマーク等各種作業環境がユーザごとに保持され，さらに変更分が格納されていることも確認できます。

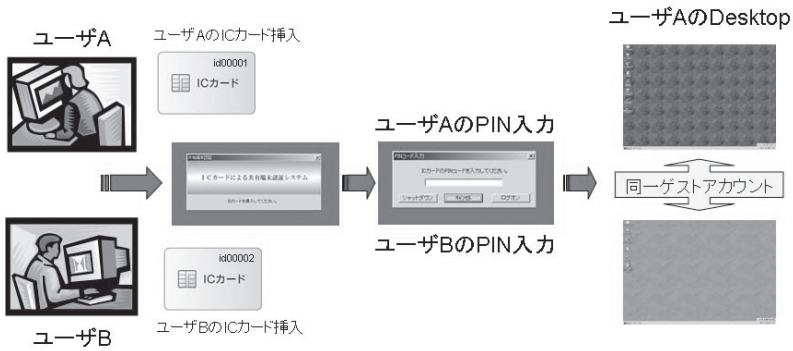

図 15　別ユーザ同一共有端末のデスクトップ比較

2．同一ユーザの遠隔地共有端末作業環境

図 16 は異なる 2 台の共有端末に同じ IC カードを挿入した場合の作業環境構築状況を確認したものです。この場合，2 台の共有端末はそれぞれ 1600 × 1200，1024 × 768 の異なる解像度を持つディスプレーを備えていますが，いずれかの異なる解像度で設定された作業環境であってもシステム側で自動的に適応することが確認できます。その他の作業環境パラメータに関してもそれぞれの変更がストレージデータとして格納されていることも確認できます。

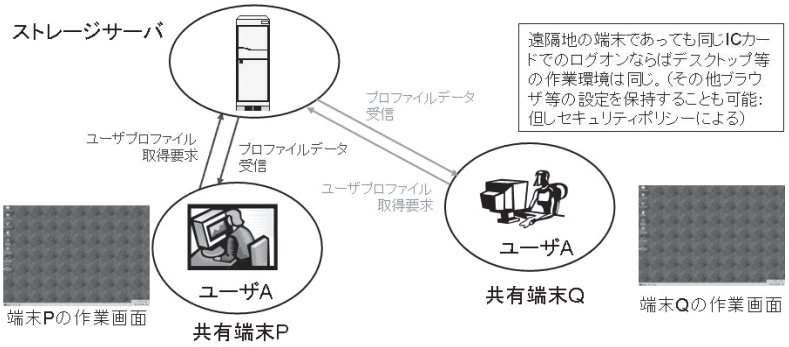

図 16　同一ユーザ別共有端末のデスクトップ比較

3．ログオン，ログオフ時間の比較

先にも述べたように, 本システムではストレージサーバからユーザプロファイルデータを取得, 格納するプロセスがログオン認証時間の遅れを引き起こすことが予想されます。通常の利用状態で, 作業環境そのものを表すプロファイル情報の容量は数百 KB から数 MB ですが, それらのデータ通信がログオン，ログオフ時間にどの程度影響するかを調べます。

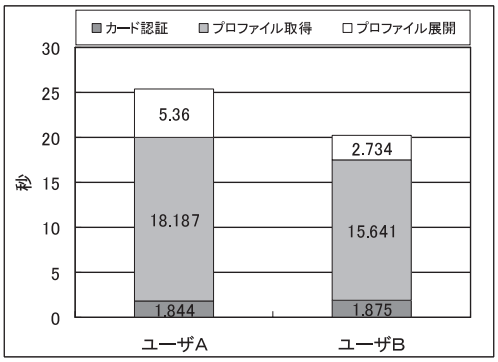

図 17　ログオンプロセスにおけるアクセス時間

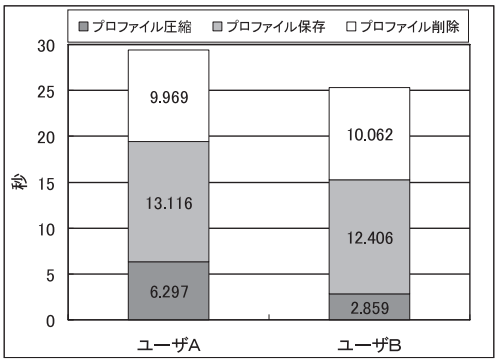

図 18　ログオフプロセスにおけるアクセス時間

図 17，18 は，それぞれログオン時とログオフ時のプロセス時間データをグラフに示したものです。それぞれは，プロファイル容量の異なる 2 枚のカードユーザとしてユーザ A（プロファイル容量≒ 9.3MB）とユーザ B（プロファイル容量≒ 7.6MB）のプロセス時間を比較しています。ここで，図 17 において，グレー（濃）は IC カード認証プロセス時間，グレー（淡）はプロファイル取得プロセス時間，白はプロファイル展開（解凍）プロセス時間を示し，図 18 において，グレー（濃）はプロファイル圧縮プロセス時間，グレー（淡）はプロファイルのストレージ保存時間，白はプロファイル削除時間を示しています。本システムにおけるネットワークスピードは共有端末 - ストレージサーバ間で約 11.5MB/ 秒（ダミーファイルの ftp 転送測定結果）です。

各プロセス時間の測定結果から，プロファイルのローカルシステムへの展開やデータ通信，データ削除などの各プロセスが作業時間へ与える影響は，ログオン，ログオフいずれにおいても無視できないことが確認できます。なお，本実験で用いたプロファイル情報は，図 7 で示したユーザプロファイル構成のすべてを対象としていますが，共有端末の利用方法をふまえて，最終的には，ストレージデータとして保存すべき情報を選択する必要があるでしょう。

Ⅶ. まとめ

共有端末利用時の利便性と経済性を考慮した利用形態として，IC カード認証と連携した非ドメイン型移動ユーザプロファイルの考え方を導入し，ドメイン構築を行うことなく，ユーザごとにどの共有端末からでも自身の作業環境が再現されるシステム構築法について解説しました。最後に紹介した実証実験では，本システムを使ってストレージサーバへのユーザプロファイルの格納と作業環境の再構築が実用的に機能していることがわかります。しかし，ユーザプロファイルの容量が増えた場合や大規模マルチユーザ環境下でのログオン，ログオフ時アクセス時間の問題，個別ユーザが所有するユーザプロファイル以外のデータの保存場所，個別ユーザのアクセスログ管理の問題，さらには GINA から CP（Credential Provider）に認証モデルを変更した Windows Vista への対応など，今後の課題はまだ多く残されています。また，本システムでは，ユーザプロファイルとストレージ格納データの紐付けを IC カードのユーザ ID 情報に基づいて行っていますが，紐付けの方法によっては生体認証との連携の可能性もあり，これも将来的な検討課題となるでしょう。

謝辞

ここで紹介した IC カードを用いた共有端末認証のための調査，研究内容は，国立情報学研究所の最先端学術情報基盤（CSI）事業の一環として行われたものです。ここに記して謝意を表します。

参考文献

[1] Zhiqun Chen, “Java Card™ Technology for Smart Cards”, Addison Wesley, 2004

[2] 葛生和人, *PKI と連携したスマートカードログオンについて－共有端末における個人認証システムへの適用－*, 名古屋大学情報連携基盤センターニュース, Vol.6, No.1, pp.27-40, 2007

[3] 葛生和人, 平野靖, 間瀬健二, 渡邊豊英, *IC カードによる共有端末認証システムの構築*, 第 35 回 コンピュータセキュリティ（CSEC）研究発表会研究報告, No.2006-CSEC-035, pp.45-50, 2006

[4] ユーザプロファイルの概念, http://www.microsoft.com/technet/prodtechnol/windowsserver2003/ja/library/ServerHelp/20f61c10-0b87-41c9-a343-b4342c5562e8.mspx

[5] 武田保真,「徹底解説 Samba LDAP サーバ構築」, 技術評論社 , 2005

[6] GINA, http://msdn.microsoft.com/msdnmag/ issues/05/05/SecurityBriefs/

[7] SID, http://support.microsoft.com/kb/243330/ja

[8] T.Okuno, “New open source CA development as Grid research platform”, http://www.naregi.org/papers/data/ggf12-caops_pki.pdf, Global Grid Forum, 2004

[9] OpenLDAP, http://www.openldap.org/

（くずう　かずと：名古屋大学情報連携基盤センター）