2006－CSEC－35 (9)
2006／12／8

ICカードによる共有端末認証システムの構築

葛生和人†, 平野靖†, 間瀬健二†, 渡邉豊英†
†名古屋大学情報連携基盤センター
〒464-8601 名古屋市千種区不老町
E-mail: †{kuzuu,hirano,mase,watanabe}@itc.nagoya-u.ac.jp

あらまし 一般に, 共有端末として使用されるPCへのログオンは, ユーザIDとパスワード入力による認証システムが採用される. しかしながら, そのような認証プロセスは, パスワードの解読のみならず漏洩の危険性も常にはらんでおり, セキュリティに対する脆弱性がしばしば指摘されてきた. PKI(Public Key Infrastructure)は, そのような情報セキュリティの脆弱性を改善する最も有効な手段の一つとして知られているが, このPKIを利用した認証システムとICカードとの連携は, 利便性がよく, なおかつ高セキュリティの個人認証システムを実現することが可能であると考えられる. 我々は, 将来的にICカードの導入と認証局の設立を視野に入れ, PKIとICカードを連携させる形で共有端末への個人認証システムを構築し, その実証実験を行った.
キーワード : セキュリティ, PKI, ICカード, 認証

Smart Card Logon for a Shared Terminal Computer based on PKI Authentication

Kazuto KUZUU†, Yasushi HIRANO†, Kenji MASE†, and Toyohide WATANABE†
† Information Technology Center, Nagoya University
Furo-cho, Chikusa-ku, Nagoya-shi, Aichi 464-8601, JAPAN
E-mail: †{kuzuu,hirano,mase,watanabe}@itc.nagoya-u.ac.jp

Abstract General authentication for a shared terminal computer such as PC is conventionally achieved with a user ID and a password. In such login authentication, however, the vulnerability is often pointed out because of the danger of password leak as well as decipherment. PKI, Public Key Infrastructure, is regarded as one of the most effective means for such information security, and we can realize a reliable and credible authentication system through combining a smart card with PKI. In consideration of both the ease of using a smart card and the safety brought about by introducing PKI, we constructed a smart card logon system for a shared terminal computer based on PKI authentication.
Keyword : security, PKI, smart card, authentication

1. はじめに

一般に, PC端末のログオンでは, ユーザIDとパスワードによる認証システムが採用されているが, このような認証手続きにおいては, パスワード解読や漏洩の危険性を常にはらむこととなる. 特に, 不特定多数のユーザを対象とする共有端末を考えた場合, 個別ユーザ管理の煩雑さが問題となり, また, そのような管理の煩雑さから認証無しにログオンが可能な状態で放置されるということさえあり得る. そのような状況を考慮すると, 共有端末においてはユーザ管理が簡便でありつつも, より強固なセキュリティの確保が望まれる. ログオン時のセキュリティに関しては, 実際, これまでもさまざまな対策が考案されており, 例えばWindows系システムではLANMAN, NTLM, Kerberos認証[1]などが採用されてきた. そのような中でPKIを用いたセキュリティ対策は現在のところ最も有効な手段の一つであることは周知の事実である. 特に, PKIを実装したICカードは, 個人認証への適用が可能であり, セキュリティと利便性の両方を兼ね備えた認証システムとして有望視されている. しかしながら, 現在のところICカードの仕様は多岐にわたり, 適用するアプリケーションの多様性ともあいまって, 日々, 変化進展するシステムやニーズに対して柔軟に対応することが困難であるという状況が生じている. 特に, 大学のような組織では, ネットワークその他のコンピュータシステムに関して, 企業とは異なる独自の管理体系を持ち, さまざまなシステム, 例えば学内無線LANやポータルサイト, シングルサインオンシステムなどとの連携を将来的に考えた場合, ICカードを利用したセキュリティシステムを自身で開発, 発展させることや開発を通じて技術的なノウハウを蓄積することは重要である. そこで, 我々は共有端末のログオン認証シス

テムを対象にPKIを導入したICカード個人認証システムの構築を試み，その実証実験を行った．

2. システム環境

本共有端末個人認証システムの構築にあたって使用しているシステム環境について説明する．先ず，OSとしては，一般的に最も普及しているWindowsを念頭に起き，特に共有端末として普及度合いの高いWindows 2000のOS環境を想定した．なお，Windows 2000（およびWindows XP）では，winlogon.exeが持つログオン管理機能を拡張するためにGINA（Graphical Identification aNd Authen-tication）が標準装備されており，この拡張APIを使うことにより独自にログオン認証プロセスを構築することができる[2]．

一方，認証システムにPKIを導入するためには，証明書を発行する認証局（CA）の構築，および認証局自身のCA証明書や失効リスト（CRL）を管理するデータベースが必要となる．そのため，本システムを構築するにあたり，Linux（Fedora Core 3）上にNAREGI-CA[3]（AICAパッケージ[4]をベースとしたグリッド向けCAサーバ）を導入してプライベート認証局を構築し，CA証明書と同時にユーザ証明書の発行，CRL作成などを行える環境を整えた．また，同システムには，ユーザID，証明書，CRLを管理するために，ディレクトリサーバとしてOpenLDAP 2.3[5]を実装している．なお，以上の環境はいずれも本格的な運用ではなく，あくまでも認証アプリの実証実験を想定したものであるため，耐タンパ性を考慮したサーバ群の構築ではなく，簡単なデスクトップレベルのOS環境に基くものとした．具体的には，表1に示すように，デスクトップPC（ThinkCentre A52T）とノートブックPC（ThinkPad Tablet）のそれぞれにWindows 2000およびFedora Core 3をインストールしOS環境を構築している．

表1 システム環境

	#01	#02
ThinkCentre A52T(IBM) (Desktop)	Windows2000 (Terminal PC)	Fedora Core 3 (CA & LDAP)
ThinkPad Tablet(IBM) (Notebook)	Windows2000 (Terminal PC)	———

3. ICカード仕様

1節でも述べたように，本認証システムの構築にあたって重要なポイントの一つは，対応するICカード仕様の多様性を，アプリケーション開発者側でできるだけ柔軟に吸収できるようにするということである．そのようなカードアプリケーション開発を考えた場合，Java Card™[6]を用いた開発環境は，カード上のアプレットを自由にプログラミング，インストールすることができ，さらにアプリケーション仕様の決定や動作確認を容易に行えるという点で非常に優れたものであると考えられる．また，ICカード上に構築されたJava Card™実行環境では，暗号化に関するAPIが標準装備されているため，PKIの導入が比較的容易であることもプログラム開発上有利である．

以上のことを考慮して，本システムの構築にあたってはJava Card™に基くカード仕様およびカードアプリ開発環境を採用することとした．さらに，カード使用時における利便性を考えて，非接触型のデュアル・インタフェースを備えたカード仕様を導入した．なお，セキュリティAPIに関してはカード内でRSA,DES,T-DESなど複数の暗号処理が可能な仕様となっている．これらカードの具体的仕様とICカードリーダライタの内容を表2に示す．

表2 ICカード（Java Card™）仕様

	準拠規格	メモリ	R/W
接触時	ISO/IEC 7816	1MB	RW4040
非接触時	ISO/IEC 14443 Type B		PD2002P

4. システム構築

ICカードを用いた共有端末認証システムの構築にあたって特徴的なことは，システム自体がさまざまな要素技術を組み合わせて成り立っているということである．それらは，端末ログオンに関する技術，PKIの実装，暗号化，証明書の扱いに関する技術，ICカード，ディレクトリサーバとの通信技術などである．いずれも既存の技術内容ではあるものの，柔軟性，発展性のあるシステムの運用を念頭に置くと，それら全てを効率的に統合し，システムを構築する必要がある．本節では，これから構築するシステムの認証手続きの流れを説明した後，それぞれのプロセスに関わる各要素技術について述べる．

4.1 認証手続きの流れ

先ず，本認証システムで行われる共有端末ログオン時の認証手続きにおいて，最終的なログオンに至るためには，4つの検証ステップを通過しなければならないものと想定した．それぞれの検証ステップとその検証プロセスのもつ意味は図1に示したとおりである．ここで示した検証プロセスは，PKIを導入したICカードから個人を特定するための必要

最低限の基本的要素と考えられる.

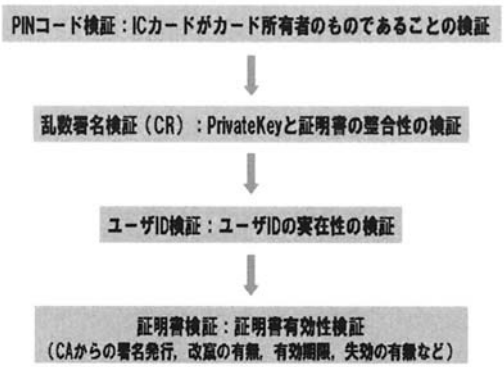

図 1 ログオン認証の検証ステップ

上の検証ステップを組み込んだ IC カードによるログオン，いわゆるスマートカードログオンの認証手続きの流れを図2に示す．ここで，IC カードには PIN コード，ユーザ証明書および私有鍵（Private Key）があらかじめインストールされており，LDAP サーバにはユーザ ID，認証局自己署名証明書，CRL が登録されているものとする．これらの情報は，IC カードリーダやディレクトリサーバと共有端末の間で行われるネットワーク通信を通して伝達されることとなる．

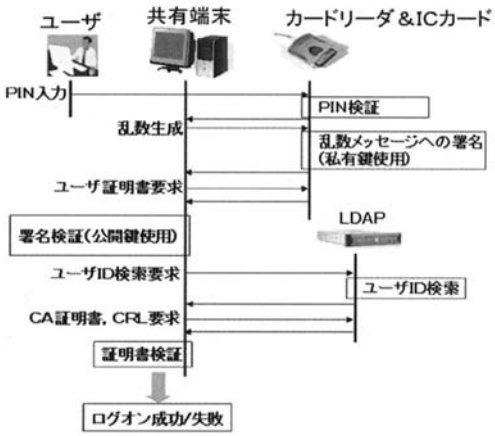

図 2 スマートカードログオン認証手続きの流れ

4．2 GINAによるログオン管理

本認証システムでは，Windows におけるログオン管理を独自に機能拡張するために，2節で紹介した GINA の API を利用することとした．実際の GINA による機能拡張は，ユーザ独自の処理関数と Windows ログオン管理プロセス（winlogon.exe）から呼び出されるコールバック関数とを組み合わせて，1つのダイナミックリンクライブラリを構成することにより得ることができる．図3は GINA の実行シーケンスを示したものである．図に示したように，本認証システムではログオン情報ダイアログボックスを表示する WlxLoggedOutSAS プロセスに，PIN コード検証，ユーザ ID 検証，証明書検証などを組み込むことでスマートカードログオン認証を実現している．なお，作成した GINA.DLL を起動するためには，表3に示したパラメータをシステムレジストリに登録する必要がある．

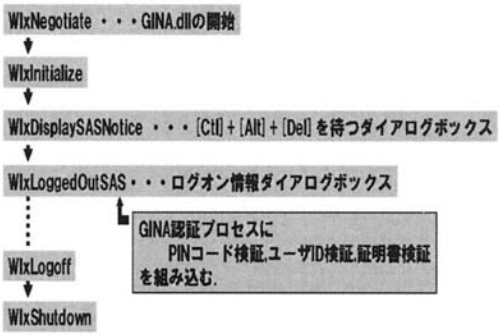

図 3 GINA 実行シーケンス

表 3 GINA.DLL のレジストリ登録パラメータ

Key	HKEY_LOCAL_MACHINES¥Software ¥Microsoft¥Windows NT ¥CurrentVersion ¥Winlogon
名前	GinaDLL
種類	REG_SZ
データ	GINA.DLL のパス

4．3 PKIの実装

IC カードに PKI を導入するにあたり，本認証システムでは，カード内に私有鍵と1枚のユーザ証明書を組み込むこととした．なお，ここでインストールする私有鍵は 1024 ビットの RSA 暗号鍵とし，あらかじめ外部（認証局）で作成したものをパスフレーズによる暗号化を解除した上で組み込んでいる．証明書に関しては，X.509 標準規格に従い，さらに ASN.1 DER フォーマットでエンコードされた単体のユーザ証明書を用いた．

以上の IC カード上の情報をもとに，カード上での署名プロセスと Windows 上での署名検証および証明書検証が PKI の認証プロセスとして実行される．

4．3．1 署名および署名検証

図1で示したログオン認証の検証ステップのうちIC カード内の私有鍵と証明書の整合性，すなわち私有鍵と証明書に含まれる公開鍵がペアをなすものであるかどうかを検証するために，本システムではチャレンジレスポンス(CR)を用いている．プロセスの概略を図4に示す．このプロセスでは，先ず共有端末側で乱数を発生させ，IC カードにその乱数データを転送する．IC カード上では，受信した乱数に対し署名処理(ハッシュ計算の後，私有鍵で暗号化)を施し，共有端末に結果を返信する．共有端末側では，IC カードよりユーザ証明書を取り出し，証明書内に含まれる公開鍵を抜き出す．そして，共有端末上で，最初に生成した乱数，受信した署名データ，および公開鍵を使用して署名検証を行う．なお，署名および署名データ検証には java.security および javacard.security クラスが用いられている．

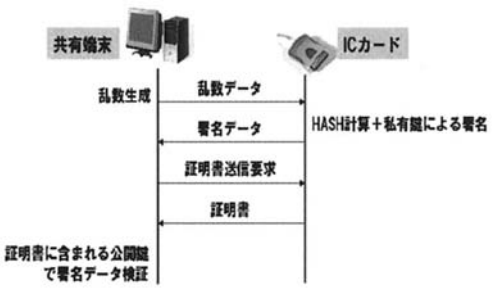

図 4 署名および署名検証プロセス

4. 3. 2 証明書検証

ユーザ証明書の検証は，図1に示したようにログオン認証ステップのうちの1つで，IC カードに組み込まれている証明書が，認証局より正しく発行されていること，改ざんされていないこと，有効期限内であること，失効していないことなどを確認する．ここでは，前節の手順で IC カードより取得したユーザ証明書に加えて，ユーザ証明書の発行元である認証局(トラストアンカ)の自己署名証明書(CA 証明書)と CRL(失効リスト)をディレクトリサーバ(OpenLDAP)よりネットワークを通じて取得する．共有端末上に取り込んだ証明書と CRL は，Windows の CSP(Crypto Service Provider)を利用して，各オブジェクトに対するコンテキストを作成した後，それらをシステムの証明書ストアに蓄積し，Crypto API を用いて証明書検証が行われる[7][8]．図5は，Crypto API を用いた証明書検証プログラムの抜粋である．なお，本システムにおいてはユーザ証明書の直上の層がトラストアンカの認証局となっているが，中間証明書が存在する場合は，各証明書をネットワークから取得し証明書ストアに蓄積することで同様の検証が可能である．

```
//証明書コンテキスト構築
pCertContext = CertCreateCertificateContext
                (X509_ASN_ENCODING, pszBuf, CertFile_Size);
//証明書ストア
CertAddCertificateContextToStore (hCertStore, pSubjectContext,
                CERT_STORE_ADD_REPLACE_EXISTING, 0) ;
・・・・・・・・・・
do {
    dwFlags = CERT_STORE_REVOCATION_FLAG |
                        CERT_STORE_SIGNATURE_FLAG |
                                CERT_STORE_TIME_VALIDITY_FLAG;
//証明書発行者検索
    pIssuerContext = CertGetIssuerCertificateFromStore (hCertStore,
                        pSubjectContext, 0, &dwFlags);
    CertFreeCertificateContext (pSubjectContext);
    if (pIssuerContext) {
        pSubjectContext = pIssuerContext;
        if (dwFlags & CERT_STORE_NO_CRL_FLAG)
        dwFlags &= ~(CERT_STORE_NO_CRL_FLAG |
                        CERT_STORE_REVOCATION_FLAG);
        if (dwFlags) break;
    }
    else if (GetLastError() == CRYPT_E_SELF_SIGNED)
        return TRUE;  //自己署名検証
} while (pIssuerContext);
```

図 5 Crypto API による証明書検証

4. 4 Java Card™アプレット

3節で説明したように本認証システムにおける IC カード仕様は Java Card™ を採用している．したがって，IC カード上に構成された Java Card™ の実行環境を利用して，PIN コード検証や署名ルーチンを独自に作成することが可能である．実際，PIN コードの検証や私有鍵を使った署名ルーチンは図6に示したようなプログラムに基いて IC カード内部処理として動作することとなり，カードに組み込まれた PIN および私有鍵の情報はカード外に抽出されることはない．このようにして作成された認証システム用の Java Card™ アプレットは，図7に示したような Java Card™ の実行環境にインストールされ，起動される[9]．

```
//PINコードの登録 (PINコードコンストラクタ)
pin = new OwnerPIN (PIN_TRY_LIMIT,  MAX_PIN_SIZE);
pin.update (bArray, bOffset, bLength);
register();
・・・・・・・・・・・・・・・・
//PINコード検証
if ( pin.check (buffer, ISO7816.OFFSET_CDATA,byteRead) == false )
                        ISOException.throwIt (SW_VERIFICATION_FAILED);
・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・
//署名アルゴリズムの指定，署名プロセスの初期化，署名
m_signature= Signature.getInstance ( Signature.ALG_RSA_SHA_PKCS1,false );
・・・・・・・・・・・・・・・・
m_signature.init ( m_priv1024_rsakey, Signature.MODE_SIGN );
・・・・・・・・・・・・・・・・
```

図 6 Java Card™ アプレット

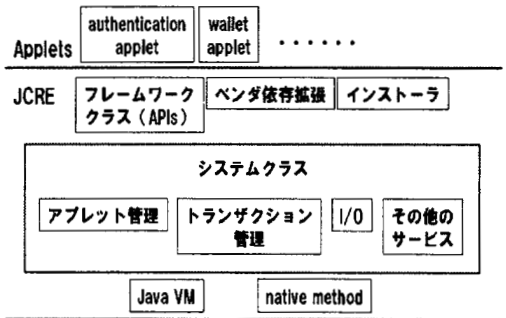

図 7 Java Card™ の実行環境

4. 5 APDUプロトコル

前節で説明した Java Card™ アプレットの起動や PIN コード検証，署名プロセスの実行は，共有端末からのリモートプロセスとなる．APDU（Application Protocol Data Units）は，このリモートプロセスを実現するための通信プロトコルであり，ISO7816-4[10]にてその仕様が定められている．具体的には，通信データはバイトコードで定義されており，図8に示したように，IC カードへ送信する Command APDU と IC カードからのデータ受信を示す Response APDU の2種類に分類される．

Command APDU structure

Mandatory header				Optional body		
CLA	INS	P1	P2	Lc	Data field	Le

Response APDU structure

Optional body	Mandatory trailer	
Data field	SW1	SW2

図 8 ISO7816-4 で規定される APDU

以上のデータ通信を実際に実現するために，共有端末側では IC カードリーダを検出できるドライバに加えて，この通信プロトコルをサポートする API が必要となる．なお，本システムで採用している IC カードリーダライタ（RW4040, PD2002P）は PC/SC 規格準拠であるため，Windows 上での IC カード用 API を利用することが可能であり，本機能を APDU 通信にも利用することとした．（図9）

4. 6 LDAPクライアント通信

4．1節で述べたように本認証システムにおける検証ステップのうちユーザ ID の検証およびユーザ証明書の検証は，ディレクトリサーバとの通信を通して行われる[11]．共有端末とディレクトリサーバ間での通信プロセスを図10に示す．ここで，ユーザ ID

```
//スマートカードコンテキストの構築
SCardEstablishContext (SCARD_SCOPE_USER, NULL, NULL, &hContext);
・・・・
//CADとの接続
SCardConnect (hContext, rsReaders [0] .szReader, SCARD_SHARE_SHARED,
                        SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1,
                        &hCard,
                        &dwActiveProtocol);
・・・・
DWORD dwRecvLength_rtrv_cert_size  = 4;
BYTE  btRecvBuffer_rtrv_cert_size [4];
BYTE apdu_rtrv_cert_size [5]  =  {0x90, 0x20, 0x10, 0x00, 0x02};
・・・・
//Command APDU送信
SCardTransmit (hCard, SCARD_PCI_T1, apdu_rtrv_cert_size,
                                    sizeof (apdu_rtrv_cert_size), NULL,
                                    btRecvBuffer_rtrv_cert_size,
                                    &dwRecvLength_rtrv_cert_size);
//Response APDU解析
LengCertificate = (DWORD) ((btRecvBuffer_rtrv_cert_size [0] <<8)
                                    +btRecvBuffer_rtrv_cert_size [1]);
```

図 9 APDU による通信プロセス

の検証には，ユーザ証明書から抽出されたユーザ ID が参照される．また，証明書有効性の検証は4．3．1節で示した手順に従う．なお，本システムでは，ディレクトリサーバとして OpenLDAP 2.3（LDAPv3 をサポート）を Fedore Core 3 上に組み込み，ユーザ ID，CA 証明書，CRL などはすべて同一の LDAP サーバデータベース上にストアしている．

図 10 共有端末－LDAP サーバ間の通信

5. 実 証 実 験

4節にて説明した共有端末での IC カードを使用した個人認証システムについて，実際のシステム，ネットワーク環境を使って実証実験を試みた．

本システムでは，2節で述べたとおり，Windows 2000 を組み込んだデスクトップPCとノートブックPCを仮想共有端末とし，さらに OpenLDAP を組み込んだデスクトップPCをディレクトリサーバとして使用する．ここでは，共有端末へのログオンプロセスを検証するため，先ず，LDAP サーバを起動し，その後，仮想共有端末である Windows 2000 を立ち上げることでスマートカードログオンを試みる．

最初に，Windows システム起動後，図11-(a)で示したとおり，IC カードの挿入要求画面が現れ，IC カード挿入すると，続いて図11-(b)に示すようにPIN コード入力が要求される．これらの画面は通常の Windows の起動画面に代わるログオン画面となる．入力は PIN コードのみで，その他，認証に必要な情報は，IC カードや LDAP サーバから自動的に取得され，検証される．

(a) (b)

図 11 スマートカードログオン画面

実験では，正規の IC カードの他，ユーザ証明書が失効している場合，ユーザ証明書の一部が改ざんされている場合，誤った PIN コード入力の場合を検証した．結果は，正規の IC カード以外はいずれもログオンできないことを確認した．また，同一のシステムを利用して，非接触リーダからも同様の動作を確認できた．なお，検証プロセスのログデータの一部を図12に示す．各検証ステップの段階でのデータ取得や，検証結果の成功/失敗の状況がログデータからも確認することができた．

```
////// PROCESS CHECK LOG ////////////
   SCardEstablishContext : SUCCESS
   SCardConnect : SUCCESS
   SCardStatus return : 0 0
      ATR = 3b e0 00 ff 81 31 fe 45 14
   PIN response : 90 00
******** PIN code verify : SUCCESS ********
   SCardTransmitSignature : SUCCESS
    4c 66 9f b8 71 04 a4 f3 47 f5 3e 98 9c
     . . . . .
    -------- EE Certificate --------
    30 82 02 32 30 82 01 9b a0 03 02 01 02
     . . . . .
******** Signature Verify : SUCCESS ********
    UID analysis from Certificate
       User ID        : kuzuu 17
    dn: uid=kuzuu, ou=People, dc=Sample, dc=net
******** User kuzuu : FOUND in LDAP ********
    dn: cn=Admin, ou=PrivateCa, dc=Sample, dc=net
    Issuer Name        : JP, Kazuto Kuzuu
    Subject Name       : JP, Kazuto Kuzuu
******** Verify Certificate : SUCCESS ********
```

図 12 各認証プロセスの検証ログ

6. まとめ

共有端末ログオン時のセキュリティ向上のため，IC カードと PKI を組み合わせた個人認証システムを構築した．本認証システムの構築にあたっては，共有端末は Window 2000 を想定し，また，IC カードアプリは Java Card™ の実行環境を利用している．本認証システムは，実証実験を通して，PIN コード検証や証明書，ユーザ ID 検証など各検証プロセスの作動状況を確認，最終的に，PKI を導入したスマートカードログオンによる個人認証システムが実現されたことを確認した．今後は，コンピュータロック時の動作など付随機能の追加に加え，Linux 等他のオペレーションシステムへの適用を目指していく．

謝 辞

本研究は，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ある．ここに記して謝意をあらわす．

文 献

[1] Richard E. Smith,「認証技術－パスワードから公開鍵まで－」，オーム社, 2003

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

[3] 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

[4] 若山公威，奥野琢人，岩田彰，村瀬晋二，鈴木春洋，「暗号ライブラリと認証局パッケージの開発」，情報処理学会 第 59 回(平成 11 年後期)全国大会講演論文集(3) pp.395-396

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

[6] Java Card™, http://jp.sun.com/products/software/consumer-embedded/card/

[7] CryptoAPI, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncapi/html/msdn_cryptapi.asp

[8] John Viega and Matt Messier,「C/C++ セキュアプログラミングクックブック」, vol.3, pp.234-238, オライリー・ジャパン, 2005

[9] Zhiqun Chen, "Java Card™ Technology for Smart Cards", Addison Wesley

[10] ISO7816-4, http://www.ttfn.net/techno/smartcards/iso7816_4.html

[11] 土井優子編,「LDAP Super Expert」, pp.116-120, 技術評論社