LL-005

ICカードによる認証システムの構築法

齊藤祐輔† 萩原洋一‡ 櫻田武嗣‡ 並木美太郎††

東京農工大学工学部情報コミュニケーション工学科†

東京農工大学総合情報メディアセンター‡ 東京農工大学大学院共生科学技術研究部††

1. はじめに

近年、IC カードの普及により、個人認証の方式は磁気カードから非接触型の IC カードに替わりつつある。中でも FeliCa[1]に代表されるマルチアプリケーション型の IC カードは、カード内に複数の情報を個別に操作でき、定期券や電子マネーとしても利用されている。また PC 端末においても FeliCa 型の IC カードは、[2]や SmartOn[3]、NSAS[4]といった認証システムでも採用されている。

このような IC カード、特に、Felica に格納された各種情報を各種の認証アプリケーションで利用するためには、カード内の情報管理やアクセスなどは一元的な手法で行うことが望ましい。そこで、本論文では、LDAP を用いてカードへアクセスする構築法を提案する。このとき、Felica へのアクセスなどでカードへの通信を占有するという問題があった。本方式ではこれらの問題を解決し、Felica 内の情報を、各種認証アプリケーションから利用できるよう、LDAP にマップしたシステムを構築した。

2. 目標

複数の認証処理を単一の IC カードを認証の鍵として行なうことが本構築法の目標である。本論文では例として認証の鍵に FeliCa を採用し、FeliCa 内に格納した一つあるいは複数の ID とパスワードによる認証情報にアクセスし、Windows ログオン認証 HTTP 基本認証の処理を実現する。

また、将来的に学内あるいは社内などを想定した、大規模ネットワーク下での利用を可能とするように、拡張性を保持するものを目指す。

3. 全体構成

既存の認証システムは、常に IC カードの有無を監視する場合、IC カードへの通信路を一つのアプリケーションが占有してしまい、他のアプリケーションからは IC カードへの操作を行うことができなかった。そこで本構築法では、図 1 に示すように、IC カードとの通信は Authentication Server Daemon（以下 ASD）が占有して行い、各認証系から認証処理を横取りする Authentication Client Program（以下 ACP）が各々、ASD とソケットなどを用いた通信を行い、認証処理を代行する。このようなクライアント - サーバモデルを採ることによって、サーバが IC カードとの通信制御を行うことができ、通信路が占有されていて他のアプリケーションが IC カードを操作できない問題を解決している。また仮に認証の鍵となる IC カードを変更する際も、

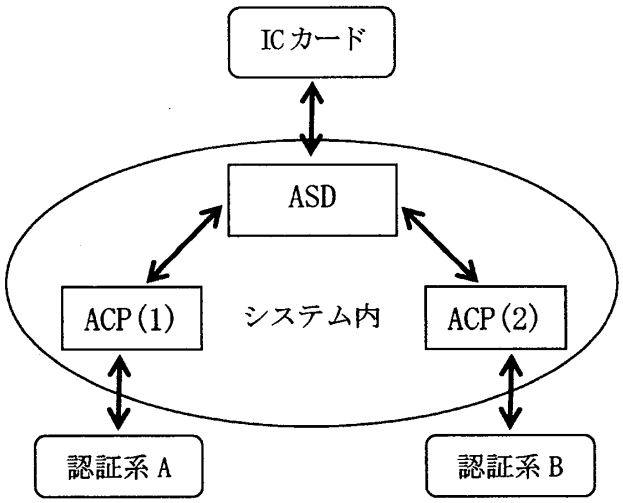

図 1. システムの全体図

A Software Architecture of Authentication Systems using an IC Card

† Yusuke SAITO, Department of Computer, Information and Communication Sciences, Tokyo University of Agriculture and Technology (TUAT)

‡ Yoichi HAGIWARA and Taketsugu SAKURADA, General Information Media Center, TUAT

†† Mitaro NAMIKI, Department of Computer, Information and Communication Sciences, Graduate School of Technology, TUAT

この構成ならばASD内のICカードとの通信モジュールのみを書き換えるだけで、他の認証の鍵への移行が可能である。

さらにICカード内の書き込み先を、サーバが制御することにより、ユーザがカード内のデータを誤って書き換えてしまうなどのトラブルを未然に回避することができる。

4. システムの設計

4.1 使用するICカードとその問題点

試作システムでは使用するICカードにFeliCaを採用する。FeliCaは一枚のカードの中で多目的のデータを管理することができ、各々のデータには個別のアクセス権を設定することが可能である。また、これによってアプリケーション間の安全な相互運用が実現されている。FeliCa内のファイルシステムは、フォルダに相当するエリアと、データに対するアクセスの種類や権限を定義するサービスによって階層状に構成されている。そしてこれらの操作を行うためのプログラミングインタフェースとして、C/C++APIライブラリ、SDK for FeliCaが提供されている。

Felicaを用いるアプリケーションは、SDKで提供されるカードへのアクセス用のAPIを用いて情報のやりとりを行う。SDKには、カードがリーダに置かれているかを判別するAPIも用意されているが、多くのアプリケーションでは、退席時やカードの利用終了時にカードをとりはずしたことを検知するために、このAPIを用いてカードの設定状況をポーリングしている。ところが、アプリケーションがポーリングを行うために、あるアプリケーションがカードを認証に用いるとそのアプリケーションがポーリングのためにカードの利用権を占有してしまい、他のアプリケーションからカードを認証に利用できなくなるという問題がある。これはカードを用いたログイン用のミドルウェアも同様である。そこで、本方式では、LDAPのプロトコルを用いると同時に、カードへのアクセスをサービスとすることで占有の問題を解決した。

4.2 プログラムの主な機能

ASDはサーバとしてACPからのメッセージを受け取り、IDやパスワードの追加、参照、変更、削除といった要求に則した操作をICカードに対して行い、結果をACPへと返答するサービスプロセスである。これによりACPには、各ICカードが持つインタフェースを意識しなくてもよいという利点が生まれる。

またASDは、ICカードのどこにユーザの各個人認証別のIDとパスワードが格納されているかを記憶しており、これらの管理とID・パスワードの入力などの操作をユーザではなくASDが行なうことで、ユーザが個人認証Aのデータと、個人認証Bのデータを誤って書き換えてしまうといったトラブルを防ぐことができる。

具体的にASDは、以下の機能を持つ。

(1) OS起動時からシステムに常駐し、ACPからの接続待ち状態にある
(2) 接続してくるACP毎に処理スレッドを生成
(3) スレッドでACPから認証系の情報を受信
(4) 認証系の情報からFeliCa内のどこへアクセスするかを検索
(5) FeliCaにアクセスしデータを操作
(6) 結果をACPに送信

以上のように、マルチスレッドによりACPを個別に処理することにより、複数のアプリケーションからのFeliCaへのアクセスを可能としている。

一方のACPは、個人認証の数だけ存在し得る。対象とする認証系に特化した手段で認証系から認証処理（ダイアログが一般的である）を横取りし、ユーザにIDやパスワードなどの入力を要求せずに、ASDに問合せを行なう。そしてASDから返された結果を認証系へと渡す。

ACPは最小限、以下の機能を満たさなければならない。

(1) 認証時にASDへ接続
(2) 認証系情報をASDへ送信
(3) IDとパスワードを受信
(4) 認証系へデータを渡す

試作システムではWindowsログオンと、HTTP基本認証をACPとして実装する。Winlogonからログオンを横取りするため、独自のGinaDLL[5]を作成する。またWebブラウザから基本認証を横取りするため、ローカルにACP機能を持つプロクシサーバを作成し、インターネットへの接続はこのローカルプロクシを介するように設定する。

4.3 プログラム間のプロトコル

クライアント-サーバ間で交わされるプロトコルにはLDAPv3[6]を採用する。LDAPは9つのオペレーションを持ち、試作システムではこの内のBind、Unbind、Add、Modify、Search、Deleteを利用し、実装する。Bind、UnbindはそれぞれASDとACP間のセッションの確立、終了を行なう

オペレーションであり、Add、Modify、Deleteオペレーションがそれぞれ認証情報の新規登録、変更、削除を行う。そして、認証時にはSearchオペレーションにより、ASDからACPへ各個人認証のIDとパスワードが渡される。

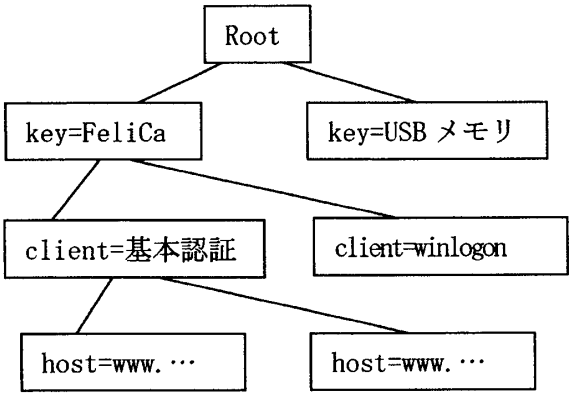

図2. ディレクトリ構成ツリー

一方、サーバが保持するディレクトリ構成ツリーは図2のようになり、クライアントの各エントリは、表1で示されるようなオブジェクトクラスに従う。表に示された属性はすべてのクライアントにおける必須属性であり、areaとservice属性の値は、FeliCa内のデータのアドレスを表している。

表1. オブジェクトクラスの例

属性名	属性値	説明
client	Winlogon	アプリケーション名
area	10	FeliCa内のデータの位置を示す値
service	1	
id	u-ske	認証情報。FeliCaに格納されている
passwd	******	

以下でSearchオペレーションを例に、処理の流れを示す。

(1) ACPはLDAPに従いBind要求をASDへと送る
(2) ASDはリーダライタを起動させ、FeliCaを探し、認識した時点でBind応答をACPへ返す
(3) ACPはアプリケーション名（client属性）を検索条件にSearch要求をASDへと送る
(4) ASDは受け取った条件でエントリを検索し、FeliCa内のどこへアクセスするかを決定し、読み出しを行い、結果をSearch応答として返す
(5) ACPがUnbind要求をASDへと送信し、セッションを終了する。

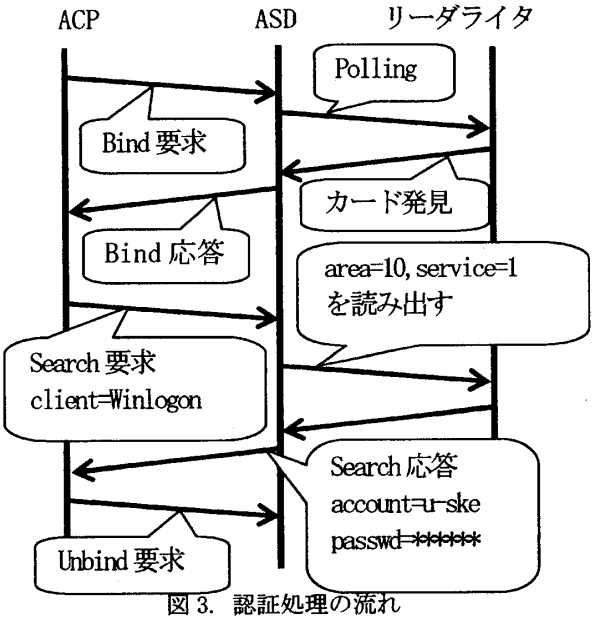

図3. 認証処理の流れ

つまり、ASDはLDAPサーバであり、ディレクトリ構造をデータ（各個人認証とそれぞれのID、パスワードといった認証情報）を保持している。このうち、ユーザのIDとパスワードは実際にはFeliCa内に格納されているため、ACP毎に割り振られたarea、service属性の値を元に、ASDがアクセスを行う。複数枚のFeliCaを用意したとすれば、ASDはそれぞれのFeliCaの同一アドレスを参照する。これにより、複数の個人によるPCの共有が可能となる。

またLDAPをプログラム間の通信プログラムとして利用することで、プロセス間通信のみならず、将来的な複数のマシン間での利用などにも容易に拡張できるといった利点が挙げられる。

5. 試作システムと評価

実装はWindows2000上で、Visual C++を用いて行った。システムの構成は図4に示す。Windowsログオンから認証処理を横取りするACP（1）と、HTTP基本認証から処理を横取りするACP（2）は、それぞれ独立してASDと通信を行なっている。ACP（1）は、FeliCaがリーダライタから外されるとコンピュータをロックするように実装したため、常にポーリングを行いFeliCaの有無を監視しているが、その間も基本認証の認証処理をACP（2）により行うことが可能となった。これは、マルチアプリケーションであるFeliCaの特長を活かすことができるようになったことを意

味する。コードの規模は、約5000行（うちASDが約3000行）程度となった。

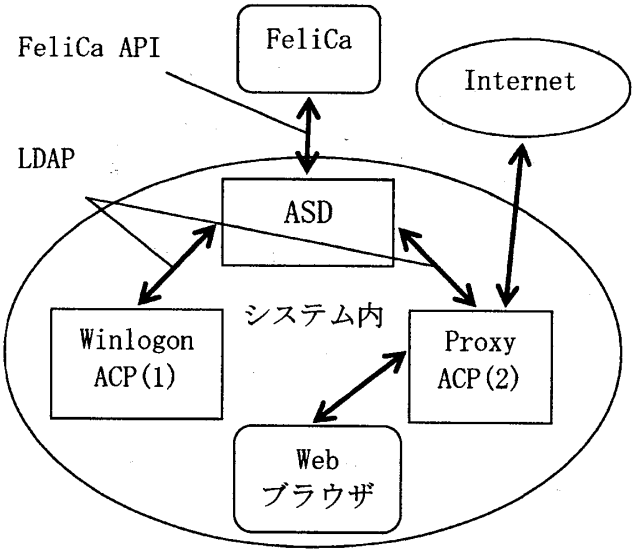

図4. 試作システムの全体図

6. おわりに

SmartOn、NSASにおいて、FeliCaでWindowsにログオンした際、他のアプリケーションからFeliCaへのアクセスができなかった問題を、本構築法では、FeliCaとの通信を専用サーバプログラムが中央管理することにより解決した。これにより、複数の認証処理を、単一のICカードを認証の鍵として行うことに成功した。また、LDAPを利用したクライアント-サーバモデルを採用することにより、認証の鍵を変更する際に掛かる労力を最小限に留めることができるようになった。今後の課題として、大規模ネットワークとの連携と、NISやPOSIXとの親和性を高めることが挙げられる。

参考文献

[1] Sony Japan | FeliCa, http://www.sony.co.jp/Products/felica/
[2] 飯塚重善, 小川克彦, 中嶌信弥:セキュアなテレワーク支援システムとシステム利用時の安心感についての考察, 情処研報, IS-89, pp.31-38(2004)
[3] Soliton - SmartOn NEO, http://www.soliton.co.jp/products/smarton/neo/
[4] 株式会社ネット・タイム NSAS, http://www.nettime.co.jp/nsas/
[5] The Essentials of Replacing MSGINA.DLL, http://www.microsoft.com/windows2000/techinfo/administration/security/msgina.asp
[6] RFC2251, www.ietf.org/rfc/rfc2251.txt