다단계 사용자 신분확인 메커니즘 설계와 구현 방안 : 출입통제 시스템 사례 중심으로☆

Toward Design and Implement to Multiple Schemes for Strong Authentication Mechanism - Case Studying : Secure Entrance System -

홍 승 필* 김 재 현**
Seng-Phil Hong Jae-Hyoun Kim

요 약

최근 유비쿼터스 컴퓨팅에 대한 관련 기술이 빠르게 발전되면서, 그와 관련 된 정보시스템 역기능(개인정보 오남용, 정보의 위험, 위협, 취약점등)의 우려 또한 증가되고 있는 추세이다. 본 논문에서는 강력한 다단계 사용자 신분확인(Multiple schemes for strong authentication) 메커니즘을 소개하고, 실제 시스템 환경에서 안정적으로 구현할 수 있는 설계 방안과 응용 방안을 제시하였다. 또한, 플랫폼에 독립적이고 다양한 응용 프로그램을 탑재할 수 있는 스마트카드(자바 카드) 기반에 암호화와 생체인식(지문)기술을 접목한 통합 사용자 인증 메커니즘을 제안함으로써, 최근 들어 중요시 되고 있는 사용자 개인정보 보호 및 공공・금융기관에서 발생될 수 있는 금융카드 도용 및 불법 복제등과 같은 위험에 대하여 체계적이고 안정적인 대안 기술을 소개하였다. 마지막으로, 스마트카드 환경 내에서 실제 시스템 개발자가 구현이 용이하고 응용이 쉬운 "스마트카드 클래스 라이브러리"를 개발하고 구체적인 자바카드기반의 생체인식 적용 방향을 제안함으로써 향후 활용방안에 대하여서도 그 방향성을 제안하였다.

Abstract

As the innovative technologies related to ubiquitous computing are being rapidly developed in recent IT trend, the concern for IT dysfunction (e.g., personal information abuse, information risk, threat, vulnerability, etc.) are also increasing. In our study, we suggested how to design and implement to multiple schemes for strong authentication mechanism in real system environments. We also introduce the systematic and secure authentication technologies that resolve the threats incurring from the abuse and illegal duplication of financial transaction card in the public and financial institutions. The multiple schemes for strong authentication mechanism applied to java technology, so various application programs can be embedded, independent of different platforms, to the smartcard by applying the consolidated authentication technologies based on encryption and biometrics (e.g., finger print identification). We also introduce the appropriate guidelines which can be easily implemented by the system developer and utilized from the software engineering standpoint of view. Further, we proposed ways to utilize java card based biometrics by developing and applying the "smartcard class library" in order for the developer and engineers involved in real system environment (Secure entrance system) to easily understand the program. Lastly, we briefly introduced the potential for its future business application.

☞ Keyword : Authentication, Smartcard, cryptography, Biopmetrics

1. 서 론

최근 유비쿼터스 컴퓨팅에 대한 관련 기술이 빠르게 발전되면서, 그에 대한 정보시스템 역기능(개인정보 오남용, 정보의 위험, 위협, 취약점 등)의 우려 또한 증가되고 있는 추세이다. 특히, 하나의 카드에 기존의 결제수단이 지녔던 기능들을 대체/보완하는 전자화폐 기능과 신용, 직불,

* 정 회 원 : 성신여자대학교 컴퓨터정보학부 교수
philhong@sungshin..ac.kr (제1저자)
** 정 회 원 : 성균관대학교 컴퓨터교육과 교수
jjhkim@comedu.skku.ac.kr
[2005/02/02 투고 - 2005/03/29 심사 - 2006/03/07 심사완료]
☆ 본 연구는 2006년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음

교통, 접근통제, 의료 등 다양한 부가가치 기능들을 통합한 스마트카드의 등장은 신용카드사용에 익숙해져 있는 일반 사용자들에게 친숙하게 받아들여져 기존 카드를 대체하거나 기업과 기관의 ID카드(개인 식별 카드)를 대체할 것으로 예측되면서, 그에 관한 우려의 목소리도 커지고 있다.

본 연구에서는 자바 언어로 구현 가능한 스마트카드 기반의 통합 사용자 신분확인 기술을 개발하고, 이를 실제 출입통제 시스템에 적용함으로써 실제 소프트웨어 개발자입장에서 활용 가능한 방안을 제시하였다. 또한 개인정보의 무결성 및 가용성 측면에서 스마트카드와 생체인식 기술의 결합을 이용하여, 개인 생체정보를 중앙 DB(데이터베이스)나 로컬 DB에 저장하지 않고 개인이 휴대할 수 있기 때문에 생체정보의 오용, 관리, 보안 문제 해결 방안을 제시하였다. 마지막으로, 개인정보의 공개여부를 사용자가 정보 입력 시 결정하거나 조절할 수 있어 프라이버시에 대한 염려를 불식시킬 수 있다.

본 논문의 구성은 다음과 같다. 1장에서는 본 논문의 개요를 소개하고 2장에서는 관련 기술과 표준화 연구 방향을 알아보았다. 3장에서는 자바기반의 스마트카드를 활용하여 생체인식 적용이 가능한 '스마트카드 클래스 라이브러리 개발 방안'에 대하여 기술하였다. 4장에서는 실제 출입 통제 시스템에 통합 사용자 인증 기술을 적용한 활용 방안을 제안 하였으며, 5장에서는 결론과 향후 연구방안을 제시하였다.

2. 관련연구

2.1 스마트카드

스마트카드는 신용카드와 같은 크기로 CPU (중앙처리장치), 메모리(EEPROM, RAM, ROM), 암호화 모듈을 내장하고 있어서 정보의 저장과 처리, 인증 및 보안기능, 외부 정보시스템과의 안전한 정보 송수신이 가능한 플라스틱 카드를 말한다[1].

스마트카드의 보안 측면에서 보면, 다양한 개인정보와 비밀키를 기억할 필요 없이 카드에 저장함으로써 상황에 따라 선택하여 서비스와 상품을 구매 및 결제하거나, 은행과의 거래, 네트웍 접근 시 인증에 사용할 수 있는 휴대용 컴퓨터라 할 수 있다. 카드 내 저장되는 데이터는 암호화되어 저장될 수 있으며, 외부와의 통신도 암호화된 상태에서 이루어 질 수 있다. 또한, 내부 데이터의 접근도 개인 식별 번호 (Personnel Identification Number)를 통한 접근제어, 인증을 통한 불법접근 제한(사용자 인증), 스마트카드 운영체제에 의한 파일접근제어 기능, 메모리 관리 유닛(MMU: Memory Management Unit)에 의한 응용프로그램간 간섭방지 기능, 카드 내 응용프로그램간 상호인증방법 구현을 통해 보안성을 높이고 있다 [1].

스마트카드는 크게 폐쇄형 플랫폼과 개방형 플랫폼으로 나뉘어 진다. 폐쇄형 플랫폼은 카드 운영체제를 개발한 회사가 카드제조, 발급, 운용에 대한 독점적 지위를 행사하는 폐쇄적 구조를 가지고 있으나, 개방형 플랫폼은 카드의 인터페이스를 공개해서 카드간 호환성이 보장되며, 카드 제조업자와 독립적으로 다양한 응용 프로그램을 개발할 수 있다. 개방형 구조의 채택은 응용 애플릿의 탑재와 삭제를 가능케 하며, 이러한 작업이 웹 기반을 통해서도 이루어질 수 있다. 그 결과 카드 내 응용프로그램의 생명 주기관리, 사용자 정보관리, 키 관리 등 카드 발급 후 사후관리 시스템(CMS: Card Management System)을 구현할 수 있는 기술이 스마트카드 솔루션의 핵심이 되고 있다.

스마트카드 시스템 개발 측면에서 개방형 플랫폼 카드가 폐쇄형 플랫폼 카드와 구별되는 점은 다음과 같다.

· COS(카드 운영체제-Card Operation System)

스펙의 공개여부
- COS에 접근할 수 있는 API의 공개여부
- EEPROM 내 애플리케이션 맵 구조 결정의 허용여부

첫째 기준은 서로 다른 카드제조업체가 제공한 카드간의 호환성을 담보하기 위한 전제 조건이다. 둘째 기준은 응용계층과 운영계층의 분리와 함께 응용계층 개발자가 COS에 대한 깊은 이해가 없어도 운영시스템 리소스에 접근할 수 있도록 함을 의미한다. 셋째 기준은 비즈니스 목적과 고객의 요구사항에 따라 카드 발급자 또는 애플릿 개발자가 파일시스템 구성과 다양한 실행코드의 인스톨을 담당할 수 있음을 의미한다.

2.2 표준화 동향 연구

사용자의 편리성, 시스템 확장성, 새로운 시장 창출의 필요성을 동력으로 발급자 중심의 카드에서 사용자 중심의 카드로 전환되고 있으며, 이러한 상황에서 다양한 사용자의 욕구를 만족시키는 카드 서비스를 제공하고, 카드 발급 후 사후 관리의 효율성을 높이기 위한 다기능 카드(Multi-Application Smart Card)가 요구되고 있다. 대표적인 다기능 스마트카드 플랫폼으로서 WFSC (Windows for Smart Card), MULTOS (Multi-Application Operating System), 자바카드가 존재하지만 개방형 플랫폼 시장은 자바카드 중심으로 움직이고 있다. MULTOS는 상대적으로 높은 보안성에도 불구하고 금융 서비스 분야에 특화 되어 있고, 특히 마스터 카드에 대한 종속성이 약점으로 작용하고 있다. WFSC는 저가이면서 윈도우 환경과의 통합이 용이하지만 보안성에 대한 검증 부족으로 시장 점유율은 저조하다. 자바카드는 플랫폼과 응용계층의 분리에서 오는 서비스 개발의 편리성과 확장성, 검증된 플랫폼의 보안성, 카드 발급 후 서비스 추가 등 다양한 장점을 제공 하고 있으며, [2] 본 논문에서는 위와 같은 이유로 자바카드 기반의 통합 사용자 인증 메커니즘을 소개하고자 한다.

2.3 자바카드

자바카드는 1996년 11월에 Schlumberger에 의해 최초로 소개되었다. Schlumberger는 스마트카드 개발기술을 연구하던 중 자바 언어가 스마트카드에 이용될 수 있다는 것을 발견하고, 자바 API(Application Programming Interface)를 개발하였고, 향후 대표적인 스마트카드 회사인 Bull과 Gemplus가 참가하여 자바카드 포럼을 결성하고 스마트카드에 자바카드 기술을 활용할 여러 가지 이슈에 대해 해결책을 찾고 있다. 선 마이크로시스템(Sun Microsystems)은 1997년 11월에 자바카드 2.0 명세서를 발표하고 1999년 3월에 버전 2.1을 발표했으며, 명세서는 크게 JCRE(Java Card Runtime Environment), JCVM (Java Card Virtual Machine), API(Application Programming Interface)로 나누어진다[3]. 명세서를 살펴보면 자바카드는 자바 기술을 스마트카드 환경에 맞게 경량화 하여 만든 것임을 알 수 있다. 자바카드는 자바를 기반으로 하고 있지만 단순히 자바 기술의 일부만을 취한 것이 아니라 카드의 특성에 맞게 트랜잭션, 애플릿 방화벽 등 기존 자바 기술에는 없는 특성들도 포함하고 있다[4, 5].

자바카드 솔루션은 물리적인 카드, 카드 애플릿, 카드와 단말기 애플리케이션과의 통신경로를 담당할 카드 리더기, 카드가 제공할 수 있는 서비스를 인식하고 해당 서비스에 접근할 수 있도록 하는 데이터 유닛간의 응용프로토콜 (APDU-Application Protocol Data Unit)을 보내는 카드 인식 단말기 애플리케이션, 프로파일 관리, 생명주기 관리, 사용자 관리, 키 관리, 발급, 애플릿 다운 등을 포괄적으로 포함하는 카드관리시스템으로 구성된다.

3. 다단계 사용자 신분확인 기술 설계 및 구현 방안

윈도우 환경 하에서 개발자는 일반적으로 스마트카드 관련 자원 관리자(Resource Manager)의 API를 통해서 카드와 통신하고, 이를 이용한 다양한 스마트카드 프로그램을 구현한다. 일반적으로, 자원 관리자 API는 사용자를 위해서 다양한 기능을 제공하고 있으나, 프로그램 개발자 입장에서는 자원 관리자를 통하여 카드 리더기와 연동 된 다양한 스마트카드 응용(DB 관리, 연결을 위한 리소스의 할당 및 반환, 통신 프로토콜 레벨의 데이터 포맷 변환 등) 기능을 실제 개발 API에서 지원하지 않는 작업을 수행해야 한다. 본 논문에서는 자원 관리자와 통신 프로토콜에 익숙하지 않은 개발자가 스마트카드사가 제공하는 기본적인 데이터 유닛간의 응용 프로토콜만을 활용하여, 통합 신분확인 기술 측면에서, 쉽게 스마트카드 프로그램을 작성할 수 있도록 SCard 클래스 라이브러리를 작성하였다.

3.1 단말기와 스마트카드 통신 메커니즘 설계

스마트카드와 단말기와의 통신 모듈 설계에서, 효과적인 통합 사용자 신분 확인 기술을 적용하기 위하여 고려해야 할 것은 다음과 같다. 첫째, 스마트카드 자원 관리자(Resource Manager)를 통해 카드 리더기와의 통신 세션을 얻은 다음 카드 리더기에 삽입된 카드와의 통신 세션을 얻는 일이며, 둘째, 일단 카드와의 통신 세션이 얻어지면 카드가 제공하는 서비스를 호출할 수 있도록 명령 APDU를 전송하는 일이며, 아래 [표 1]은 카드 내 적용되는 명령 APDU의 구성을 보여주고 있다.

단말기와 카드간 통신 프로토콜은 T=0(데이터 전송단위가 문자)과 T=1(데이터 전송단위가 블럭)로 이루어지지만 T=0이 일반적이다. T=0 타입의 전송 프로토콜(TPDU: Transport Protocol Data Unit)은 카드와 단말기의 물리적 계층, 데이터 링크 계층, 전송 계층을 통하여 전송된다. 응용계층간 프로토콜인 APDU의 수행 단계는 단말기에서 카드로의 명령어 전송, 카드에서의 명령어 처리 및 카드에서 단말기로의 응답으로 구성된다. 특정 응답은 특정 명령어와 쌍을 이루게 되며, 표 2는 응답과정에서 데이터 코드 내 APDU가 사용되는 방안을 보여주는 예제이다.

〈표 1〉 명령 APDU

Header					Body	
CLA	INS	P1	P2	Lc	Data	Le

코 드	내 용	길이(바이트)
CLA	명령어 클래스	1
INS	명령어 코드	1
P1	명령어 파라미터1	1
P2	명령어 파라미터2	1
Lc	전송되는 데이터 길이	0/1
Data	메시지(데이터)	가변
Le	응답메시지 데이터 길이	0/1

〈표 2〉 응답 APDU

Body	Trailer	
Data	SW1	SW2

코 드	내 용	길 이
Data	카드로부터의 응답 데이터	가 변
SW1	명령처리상태	1
SW2	명령처리상태	1

3.2 SCard 클래스 라이브러리 설계

SCard 클래스 라이브러리는 클래스 스마트카드를 나타내며 일반적으로 'CSC'라는 접두사를 사용한다. [표3]은 SCard 클래스 라이브러리를 보여주며, 다음과 같이 4가지로 구성되어 있다.

1. 자원 관리자의 DB와 카드리더기
2. 스마트카드 관리

3. 카드 리더기와 스마트카드의 연결 관리
4. 카드 리더기와 스마트카드의 데이터 및 통신 프로토콜 변환 담당

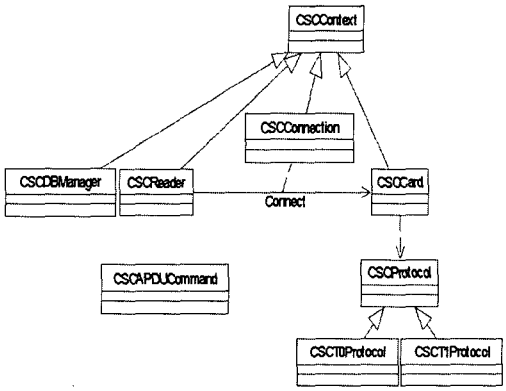
현재 구현된 클래스는 APDU 명령과 카드 리더기와 스마트카드간의 통신을 지원하는 클래스로 구성되어 있으며, 자원 관리자의 DB관리, 카드 리더기 또는 스마트카드에서 발생하는 이벤트 처리 등을 수행하는 클래스 라이브러리로 구성되어 있다. 그림 1은 SCard 라이브러리의 구성이 어떻게 설계되어 있는지를 보여주고 있으며 각각의 클래스에 대한 설명은 아래와 같다.

<표3> SCard 클래스 라이브러리

구분	역할	담당 클래스
· 지원관리자 · 카드리더기 · 스마트카드	리소스관리 장치 이벤트처리	CSCDBManager CSCReader CSCCard
카드리더기와 스마트카드 연결관리	카드리더기, 스마트카드의 클래스 생명주기 관리	CSCConnection
통신처리	APDU 커맨드 구성 응답해석 통신 프로토콜 데이터 변환	CSCAPUDCommand CSCProtocol CSCT0Protocol

다음은 Scard 클래스 구조 관련 주요 역할에 대하여 설명한 것이다.

- CSCContext: 자원 관리자의 Context handle을 관리하는 클래스로 각 클래스가 초기화될 때 Context handle을 공유하는 역할 수행
- CSCDBManager: 자원 관리자의 카드 리더기, 스마트카드 등록, 수정, 삭제의 관리기능 수행
- CSCConnection: CSCReader와 CSCCard를 초기화시키고 카드 리더기와 스마트카드를 연결시키는 역할 수행
- CSCReader: 카드 리더기의 상태 확인 및 이벤트 처리



〈그림 1〉 SCard 클래스 구성도

- CSCCard: 스마트카드를 통한 APDU전송, 카드의 ATR정보 및 상태 처리
- CSCAPDUCommand: 애플리케이션 레벨의 스마트카드 커맨드 생성관리 시 유틸리티 클래스로 사용
- CSCProtocol: 전송 프로토콜의 추상 클래스로CSCCard에서 데이터 통신 시 적절한 프로토콜을 선택하는 역할수행
- CSCT0Protocol: 통신 프로토콜 중 T=0의 데이터 포맷 변환 및 자료처리 수행
- CSCT1Protocol: 통신 프로토콜 중 T=1의 데이터 포맷 변환 및 자료처리 수행

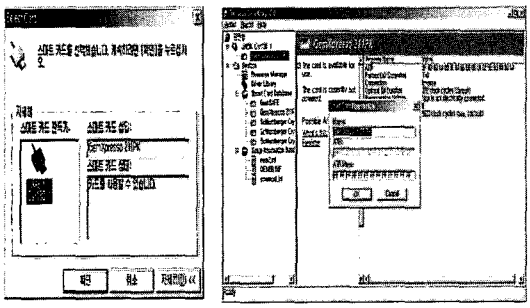
3.3 Scard 클래스 형상 관리 및 구현 방안

SCard 클래스 라이브러리는 DLL형태로 배포되며 정적 링크로 DLL에서 참조한 라이브러리 루틴을 포함하므로 크기는 커지지만, 프로그램 개발 시 별도의 다른 라이브러리를 포함하지 않아도 동작하도록 구성하였다.

SCard 클래스 라이브러리의 시작은 카드리더기와 스마트카드를 연결하는 부분으로부터 시작한다. 먼저, CSCConnection 클래스를 생성하고, UIDlgSelectCard()를 호출하여 카드리더기와 스마트카드를 선택한다. UIDlgSelectCard()가 호출

되면 카드리더기와 카드리더기에 삽입된 스마트카드가 나타난다. 이때, UIDlgSelectCard()는 연결하려는 카드리더기와 스마트카드가 자원관리자 DB에 사전에 등록되어 있어야 하며, 카드리더기는 드라이버를 설치하면서 자원관리자 DB에 등록이 된다.

본 논문에 적용된 개발 카드 환경은 Gemplus 사에서 만든 'SmartDiag'라는 프로그램을 이용하여 등록되지 않은 스마트카드를 그림 2와 같이 등록하도록 구현했다.



〈그림 2〉 Gemplus SmartDiag v2.0을 이용한 스마트카드 등록화면

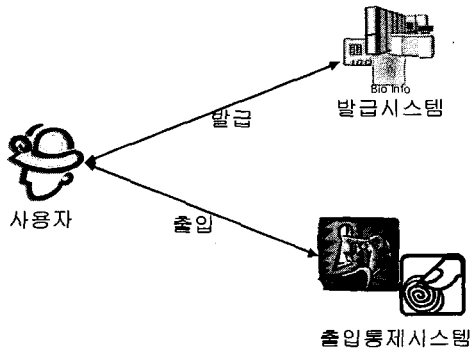
카드리더기와 스마트카드가 정상적으로 연결되었다면 CSCConnection에서 생성한 CSCCard의 인스턴스를 얻을 수 있다. 즉 UIDlgSelectCard()의 반환 값이 NULL이 아니게 된다. 카드리더기와 스마트카드의 연결이 성공적으로 진행된 후에는 APDU 명령를 통해서 스마트카드로 명령어를 보내게 된다. APDU 커맨드를 보낼 때는 CSCAPDU Command로 APDU Command를 생성하고, 생성된 APDU커맨드를 CSCCard의 sendCommand()를 호출하여 보내면 된다.

4. 출입통제 시스템 설계 및 구현

4.1 출입 통제 시스템 설계

3장에서 개발된 Scard 클래스 라이브러리의 활용방안을 제시하기 위하여, 지문인식과 암호화를 적용한 통합 사용자 인증기술을 실제 출입통제 시스템에 적용하여 보았다. 실제 개방형 플랫폼 스마트카드 기술 가운데 사용자 단말과 밀접한 자바카드 애플릿과 애플리케이션 구현을 통하여 아래 그림 3과 같이 출입통제 시스템 아키텍처를 설계하였다.

본 논문에서의 자바카드는 단말기 측의 요구에 따라 개인 식별 기능, 지문 데이터 저장 기능, 인증서 저장 기능을 구현했으며, 특히 지문을 이용한 출입통제 시스템에 활용되도록 설계하였다.



〈그림 3〉 지문인증과 암호화를 이용한 출입통제 시스템 구성도

지문인식과 암호화를 이용한 통합 사용자 인증기반의 출입통제 시스템은 크게 출입통제 부문과 카드발급 부문으로 구분되어 진다.

사용자는 발급시스템으로부터 자신의 지문이 저장된 스마트카드를 발급 받고 그것을 이용하여 지문으로 출입이 통제되는 시스템에 접근하는 것이다. 이 지문을 이용한 출입통제 시스템의 단계는 크게 두 부분으로 이루어져 있다. 그 첫 번째는 발급 단계이다. 사용자는 자신의 ID카드를 발급 받을 때, 발급 시스템에 가서 자신의 지문을 등록을 하게 되며, 이 사용자의 지문은 사용자 자신의 ID카드에 저장되게 된다. 그러나

별도의 DB에는 사용자의 지문이 저장되지 않음으로써 사용자는 자신의 생체정보에 대한 누출의 고민 없이 자신의 ID카드를 사용할 수 있다.

두 번째 단계는 출입 단계이다. 사용자는 출입이 제한된 지역으로 출입하고자 할 때, 출입문에 부착된 카드리더기에 자신의 ID 카드를 꽂고, 지문인식기를 통하여 등록한(자신의 ID카드에 저장된) 자신의 지문을 인식시킨다. 출입통제 시스템은 지문인식기로부터 인식된 지문정보 A와, 스마트카드로부터 읽혀진 지문정보 B를 비교하여, 동일하면 출입을 허가하지만, 그렇지 않다면 출입을 허가하지 않는다. 이때 출입 통제 단계에서, 지문의 비교는 단순히 ID 카드의 소유자가 사용자가 맞는지를 확인하는 것으로 생각할 수 있다. 그러나 ID 카드에는 사용자 정보가 들어가 있음으로, 그 사용자 정보를 이용하여 출입 허용의 대상이 되는 지를 확인할 수가 있으며, 또한 로그 파일을 이용하여 출입의 기록을 남길 수가 있다. 또한, 지문 정보를 이용함으로써 카드의 도난 및 오용을 막을 수 있다.

4.2 지문 인증을 이용한 출입통제 프로세스

출입통제 프로세스는 다음의 세 가지 모드로 개발되었다.

- 비 암호화 데이터 모드 : 지문 정보가 스마트카드로부터 출입통제 시스템으로 보내어 질 때, 암호화가 되지 않고 보내진다.
- 공유키 암호화 데이터 모드 : 지문 정보가 스마트카드로부터 출입통제 시스템으로 보내어 질 때, 이미 기 공유된 키를 가지고 암호화되어 보내어 진다.
- 세션키 암호화 데이터 모드 : 지문 정보가 스마트카드로부터 출입통제 시스템으로 보내어 질 때, 출입통제 시스템은 세션키를 생성하여 스마트카드와 공유를 한다. 지문정보는 이 세션키에 의해서 암호화되어 보내어 진다.

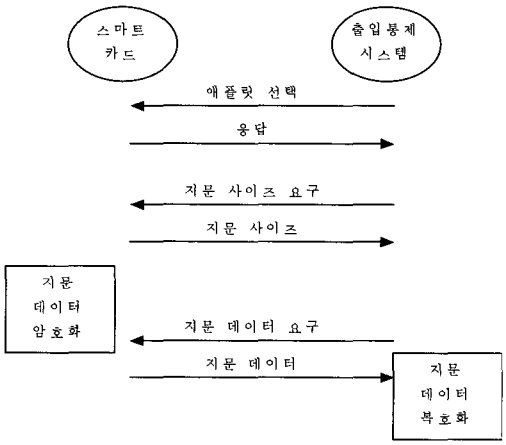
4.2.1 비 암호화 데이터 모드

비 암호화 데이터 모드는 스마트카드와 출입통제 시스템간의 연결 채널이 안전하다는 가정에서 사용 가능한 모드이다. 비 암호화 데이터 모드의 프로토콜은 다음과 같다.

(1) 출입통제 시스템은 스마트카드에 설치되어 있는 애플릿 중에 지문 처리하는 애플릿의 AID를 사용하여 애플릿을 결정한다.
(2) 출입통제 시스템은 스마트카드에 저장되어 있는 지문 데이터의 사이즈를 요구한다.
(3) 스마트카드는 저장된 사용자의 지문 정보의 크기를 출입통제 시스템에 알려준다.
(4) 출입통제 시스템은 스마트카드에 사용자의 지문 정보의 사이즈에 따라, 지문 데이터를 요구하게 된다.

4.2.2 공유키 암호화 데이터 모드

그림 4는 공유키 암호화 데이터 모드의 프로토콜을 보여주고 있다.



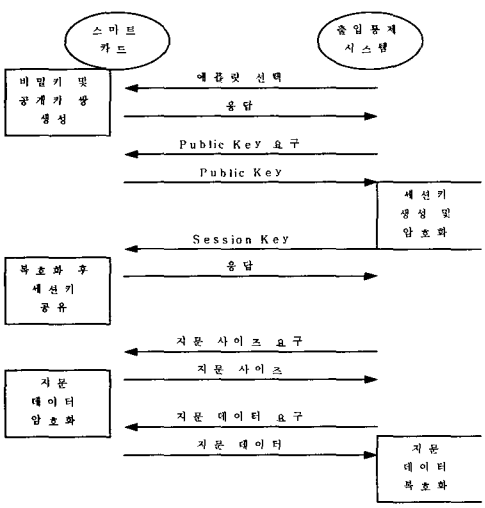
〈그림 4〉 공유키 암호화 데이터 모드의 프로토콜

공유키 암호화 데이터 모드는 스마트카드와 출입통제 시스템간의 연결 채널이 안전하지 않으며, 상호간에 비밀키를 공유하고 있다고 가정

한 상태에서 사용하는 모드이며, 공유키 암호 모드의 프로토콜은 다음과 같다.

(1) 출입통제 시스템은 스마트카드에 설치되어 있는 애플릿 중에 지문 처리하는 애플릿의 ID를 사용하여 애플릿을 결정한다.
(2) 출입통제 시스템은 스마트카드에 저장되어 있는 지문 데이터의 사이즈를 요구한다.
(3) 스마트카드는 저장된 사용자의 지문 정보의 크기를 출입통제 시스템에 알려준다.
(4) 출입통제 시스템은 스마트카드에 사용자의 지문 정보의 사이즈에 따라, 지문 데이터를 요구하게 된다.
(5) 스마트카드는 지문 정보를 보내기 전에 기 공유된 암호화키를 이용해서 지문 정보를 암호화 한 후 출입통제 시스템에 보낸다.
(6) 출입통제 시스템은 받은 지문 정보를 기 공유된 암호화키를 이용해서 복호화 한 후 사용자의 지문 정보를 얻는다.

4.2.3 세션키 암호화 데이터 모드



〈그림 5〉 세션키 암호화 데이터 모드의 프로토콜

세션키 암호화 데이터 모드는 스마트카드와 출입통제 시스템간의 연결 채널이 안전하지 않으며, 상호간에 비밀키를 공유하고 있지 않기 때문에 공개키 암호 시스템을 이용해서 세션키를 생성한 후 지문 정보를 암호화해서 전송하는 모드이다. 이 방식은 공개키 기반 구조(PKI)를 가정한 환경에서 적용 가능하다. 세션키 암호화 데이터 모드의 프로토콜은 그림 5과 같다.

(1) 스마트카드에 지문 처리 애플릿이 설치 될 경우 자동적으로 공개키와 개인키 쌍이 생성된다.
(2) 출입통제 시스템은 스마트카드에 설치되어 있는 애플릿 중에 지문 처리하는 애플릿의 AID를 사용하여 애플릿을 결정한다.
(3) 출입통제 시스템은 스마트카드에 공개키를 요구한다.
(4) 스마트카드는 기 생성시킨 키 쌍 중 공개키를 보내준다.
(5) 출입통제 시스템은 세션키를 생성한 후, 공개키로 암호화하여 보낸다.
(6) 스마트카드는 받은 암호화된 세션키를 자신의 개인키를 사용하여 복호화하여 세션키를 얻어낸다.
(7) 출입통제 시스템은 스마트카드에 저장되어 있는 지문 데이터의 사이즈를 요구한다.
(8) 스마트카드는 저장된 사용자의 지문 정보의 크기를 출입통제 시스템에 알려준다.
(9) 출입통제 시스템은 스마트카드에 사용자의 지문 정보의 사이즈에 따라, 지문 데이터를 요구하게 된다.
(10) 스마트카드는 지문 정보를 보내기 전에 상호간에 공유된 세션키를 이용해서 지문 정보를 암호화 한 후 출입통제 시스템에 보낸다.
(11) 출입통제 시스템은 받은 지문 정보를 세션키를 이용해서 복호화 한 후 사용자의 지문 정보를 얻는다.

4.3 출입통제 시스템 구현

4.3.1 지문인식 애플릿 구현 방법

지문 애플릿을 개발하기 위해서 jdk1.2.2, java_card_kit-2_1_2, GemXpresso III[6, 7]을 사용하여 아래와 같이 구현했으며 수행하는 주요 기능은 다음 그림 6과 같다.

```
public void process(APDU apdu) throws ISOException {
    byte[] apduBuffer = apdu.getBuffer();
    if(apduBuffer[ISO7816.OFFSET_CLA] == CLA_FINGER)    {
        switch(apduBuffer[ISO7816.OFFSET_INS]) {
            case INS_STOREFINGER: // 지문 저장
                storeFinger(apdu);      break;
            case INS_GETENDFINGER: // 지문을 단말기로 내보낸다
                getFinger(apdu);        break;
            case INS_DELETEFINGER: // 지문을 삭제
                deleteFinger(apdu);     break;
            case INS_GENRSAKEY: // RSA public key를 내보낸다
                genRSAKey(apdu);        break;
            case INS_PUTDESKEY: // 단말기로부터 DES key를 받아 저장
                putDesKey(apdu);        break;
            default :

    ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
                break;
        }
    }else if (apduBuffer[ISO7816.OFFSET_CLA] == ISO7816.CLA_ISO7816) {
        switch(apduBuffer[ISO7816.OFFSET_INS]) {
            case ISO7816.INS_SELECT:
            break;
        }
    }
    else ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
}
```

〈그림 6〉 코드 구현

- process() : 단말기로부터 들어오는 명령 APDU를 해석하여 그에 따른 함수를 호출한다. 지문 애플릿에서 정의한 명령 APDU는 INS_STOREFINGER, INS_GETEFINGER, INS_DELETEEFINGER, INS_GENRSAKEY, INS_PUTDESKEY 가 있다.
- storeFinger() : 지문 데이터를 저장하라는 APDU가 왔을 때 process 메서드를 통해 호출되는 함수이다. 지문 데이터는 APDU 버퍼 크기보다 크므로 여러 번의 명령 APDU를 통해 지문 데이터를 저장할 수 있다. 지문을 저장하는 APDU 가 올 때 그 APDU 중 P1의 값을 먼저 체크하여 P1 이 01 이면 들어오는 데이터는 지문의 전체 길이로 저장하고 02 또는 03인 경우 지문데이터에 저장한다.
- deleteFinger() : 지문 데이터를 카드에서 삭제하라고 하는 APDU가 왔을 때 process 메서드를 통해 호출되는 함수이다. EEPROM에 잡혀있는 지문데이터 배열을 00으로 채우고 길이를 0으로 설정해 주기만 하면 된다.
- getFinger() : 지문 데이터를 보내달라고 하는 APDU가 왔을 때 process 메서드를 통해 호출되는 함수이다. 지문 데이터는 APDU 버퍼 크기보다 크므로 여러 번의 응답 APDU 를 통해 단말기로 보낸다. 명령 APDU의 P1의 값을 먼저 체크하여 P1 이 01 이면 지문 데이터의 전체 길이로 내보낸다. 02 인 경우는 지문 데이터를 DES 키를 사용해 암호화하여 내보낸다. 여기서 사용되는 DES 키는 카드와 단말기 사이에 이미 공유되어있는 키이다. 이 키 공유에 관한 것은 putDESKey() 함수를 통하여 이루어진다. 03 인 경우는 지문 데이터를 암호화 하지 않고 그대로 내보낸다.
- getRSAKey() : 단말기에서 RSA 공개키를 요구하는 APDU 가 왔을 때 process()를 통해 호출되는 함수이다. RSA 공개키는 애플릿이 설치 될 때 이미 생성되어 있다.
- putDesKey() : 지문 데이터를 암호화하여 내보내기 위해서 DES 키를 단말기와 카드가 공유해야 한다. 이를 위해 단말기 측에서 DES 키를 보내는 명령 APDU가 왔을 때 호출되는 함수이다. 단말기에서 DES 키를 보

낼 때 그냥 보내는 것이 아니라 카드의 RSA 공개키를 사용해 DES 키 값을 암호화하여 보낸다. 이 애플릿에서는 이렇게 카드에서 보내는 DES 키를 사용하여 암호화하는 방법과 이미 애플릿에 존재하는 DES 키를 사용하여 암호화하는 방법 두 가지를 구현하였다.

4.3.2 지문 인식 어플리케이션 구현 방안

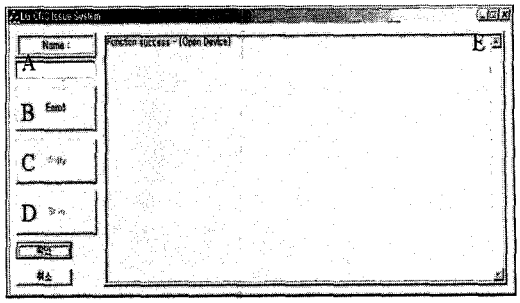
상세 프로그램은 시나리오 상에서 발급 단계와 출입 단계로 나누어진 부분이다. 뒤이어 기술된 구현환경에서 보듯이 이 프로그램은 PC 에서 개발되었다. 본 프로그램은 다음과 같은 환경을 가지고 제작되었다.

- OS: Windows 2000
- 프로그래밍 언어: Visual C++6.0
- 암호 라이브러리: Microsoft CryptoAPI (CAPI) 2.0 및 openssl 0.9.5a
- 스마트카드: Gemplus사의 자바카드 GemXpress 211PKIS
- 지문 인식기와 라이브러리 : SecuGen사의 지문인식기[8]와 NitGen사의 SecuBSP[9]

4.3.3 발급 시스템 구현 방안

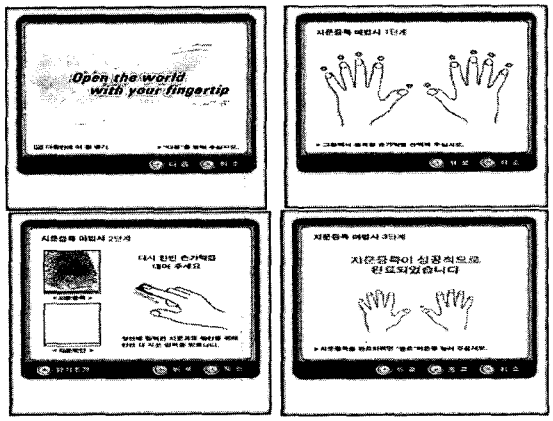
발급 시스템은 사용자의 지문을 인식하여 스마트카드에 저장하여 발급하는 시스템이다. 지문 인식 부분에 SecuBSP 라이브러리를 사용했으며, 스마트카드에 관련한 라이브러리로 자체 제작한 SCard 라이브러리를 사용했다. 시스템이 시작되면 각각의 변수들에 초기 값들이 설정되고, 설정된 변수들을 이용하여 지문 인식 장치를 오픈하게 된다. 발급 시스템의 시작 화면은 다음 그림 7과 같다.

E에서 보여 지는 "Function success - [Open Device]"는 지문인식 장치를 오픈 하는데 성공적으로 수행하였다는 메시지이다. A부분은 사용자의 이름 또는 유일한 ID를 넣는다. 이곳에 기



〈그림 7〉 발급시스템 시작 화면

입된 정보는 지문 정보와 함께 스마트카드에 저장된다. 또한, 지문을 인식하였을 경우 이 곳에 기입된 정보가 나타나게 된다. 이 곳에 기입되는 사용자 정보는 자동적으로 저장된다. B 부분 Enroll은 사용자의 지문을 입력 받는다. 그림 8은 단계별로 지문을 입력받는 장면이다.



〈그림 8〉 단계별 지문입력

4.3.4 출입 통제 시스템 구현 방안

출입통제 시스템은 사용자가 제시하는 스마트카드 내에 저장된 지문과 사용자의 지문을 서로 비교하여 동일한 지문으로 판명될 경우 출입을 허가하는 시스템이다. 출입을 통제하는 장소의 안전성에 따라 - 스마트카드와 출입통제 시스템 간의 데이터 교환 시 공격의 가능성 여부 - 스마트카드 내에 저장되어 있는 지문이 평문으로 출입통제 시스템으로 전달되는 비암호화 데이터

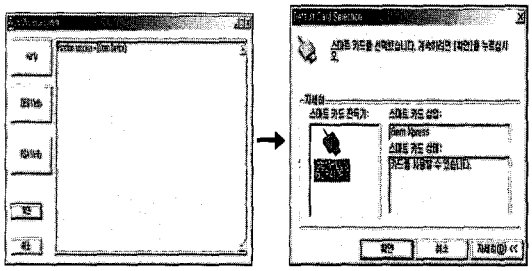
모드와 스마트카드와 출입통제 시스템 간에 미리 공유되어 있는 키를 가지고 암/복호화를 행하는 공유키 암호화 데이터 모드, 마지막으로 세션키를 생성하여 그 세션키를 두 객체가 공유를 하고, 그 세션키를 이용하여 암, 복호를 수행하는 세션키 암호화 데이터 모드로 구분하였다. 이 세 가지 모드를 하나의 프로그램을 보여주기 위하여 각각을 순차적으로 "Verify", "DES Verify", "RSA Verify" 버튼으로 구성하였으며, 이는 그림 9에서 확인할 수 있다.

- Verify: 비암호화 데이터 모드
- DES Verify: 공유키 암호화 데이터 모드
- RSA Verify: 세션키 암호화 데이터 모드

출입통제 프로그램에 필요한 라이브러리는 다음과 같다.

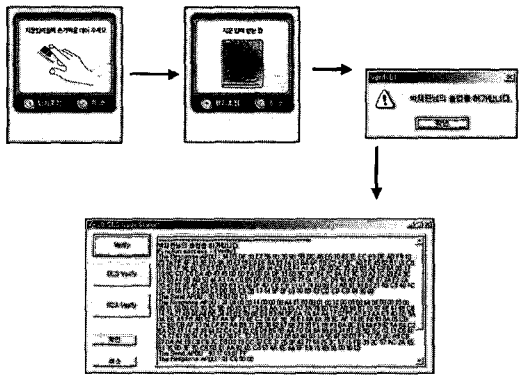
- 지문인식 라이브러리: SecuBSP 라이브러리
- 스마트카드: GemXpress 211PKIS 카드, 카드와의 통신을 위한 SCard 라이브러리
- 암호 라이브러리
  - "DES Verify"를 수행하기 위하여 MS 사의 CryptoAPI(CAPI) 사용
  - "RSA Verify"를 수행하기 위하여 openssl 0.9.5a 버전을 사용

그림 9는 출입통제 프로그램을 수행하였을 경우 보이는 첫 번째 메인 화면이다.



〈그림 9〉출입통제 메인 화면 및 카드 선택

위의 메인 화면에서 "Verify", "DES Verify", "RSA Verify" 버튼을 선택하였을 경우 사용자에게 보이는 화면은 동일하게, 카드를 선택하는 그림으로 넘어간다. 그림 10은 카드를 선택 후 사용자의 지문을 인식 받는 그림이며, 인식한 사용자의 지문과 사용자가 제공하는 스마트카드에 저장되어 있는 지문이 일치한 경우, 지문의 소유주의 이름이 팝업(Pop-Up) 윈도우로 생성된다. 일련의 검증 과정이 끝났을 경우 화면은 각각 보내고 받았던 APDU 및 수행하였던 함수들의 성공 여부 등의 로그를 보여준다.



〈그림 10〉 사용자 지문 인증

5. 결론 및 제언

본 연구에서는 개방형 플랫폼 환경 하에서 강력한 인증을 제공하기 위해서 자바카드 기반에서 암호화와 지문 인증을 결합한 통합인증 시스템을 구현했으며, 이를 출입통제 시스템에 적용함으로써, 스마트카드 관련 개발자 측면에서, 자바카드관련 응용 프로그램을 개발하기 편리하도록 SCard 클래스 라이브러리를 구현하였다. 특히 스마트카드 관리 시스템 환경 내 적절한 보안성을 판단하고 이를 구현, 관리해야 하는 보안 담당자, 운영자 측면에서 국제 표준을 이용한 통합 사용자 인증 구현 방안에 대하여 구체적인 가이드라인을 제시함으로써, 학술적 연구가 실제

현업에서 활용 될 수 있도록 방향성을 제시 하는데 주력하였다. 향후 유비쿼터스 환경 내 개인정보 보호 측면에서 IC 카드 기반의 개인정보 오.남용 방지 메커니즘 개발과, 센서기술(RFID 보안), 모바일 환경 내 개인정보보호 메커니즘 활용 방안 등에 대한 연구에 주력 할 계획이다.

참 고 문 헌

[1] W.Rankl & W. Effing, "Smart Card Handbook 2nd Ed.", Wiley, 2000.

[2] Jose Luis Zoreda, Jose Manuel Oton, "Smart Card", Artech House, 1994.

[3] 홍승필, 김명철, "정보보호의 이해", 2004.

[4] Sun Microsystems Inc., "Java Card V2.2 Development Kit Users Guide".

[5] A. W. Dent and C. J. Mitchell, "User's Guide to Cryptography and Standards", Artech House, 2005.

[6] Q. Tang and C. J. Mitchell, "Cryptanalysis of two identification schemes based on an ID-based cryptosystem" IEEE Proceedings on Communications, 2005.

[7] Sun Microsystems Inc., "Programming with Java Card Technology SL340 Student Guide", 2001.

[8] Zhiqun Chen, "Java Card Technology for Smart Cards", Sun Microsystems Inc., 2000.

[9] Gemplus, "GemXpresso RAD III User Guide V3.2", 2002.

◐ 저 자 소 개 ◑

홍 승 필 (Seng-Phil Hongi)

1993년 Indiana State University (학사)
1994년 Ball State University (석사)
1997년 Illinois Institute of Technology (박사수료)
2002년 한국정보통신대학교 (박사)
1997년 ~ 2004년 LG CNS Systems, Inc.
2005년~현재 성신여자대학교 컴퓨터정보학부 전임강사
관심분야 : 접근제어, 통합인증, 정보보호 아키텍처, 유비쿼터스 보안, 프라이버시
E-mail : philhong@sungshin.ac.kr

김 재 현 (Jaehyoun Kim)

1988년 성균관대학교 수학과 졸업(학사)
1992년 Western Illinois University 대학원 전산학과 졸업(석사)
2000년 Illinois Institute of Technology 대학원 전산학과 졸업(박사)
2001년~2002년 국민은행(구 주택은행) CTO
2002년~현재 성균관대학교 컴퓨터교육과 조교수
관심분야 : 객체지향 소프트웨어공학, 컴포넌트 기반 개발(CBD), etc.
E-mail: jhkim@comedu.skku.ac.kr