IC카드를 위한 Soft masking 방법의 설계 및 구현

전 용 성, 주 홍 일, 전 성 익
한국전자통신연구원
전화 : 042-860-5855 / 핸드폰 : 017-503-9406

Design and Implementation of Soft masking method for IC card

Yong-Sung Jeon, Hong-Il Ju, SungIk Jun
Electronics and Telecommunications Research Institute
E-mail : ysjeon@etri.re.kr

Abstract

Soft masks mean that part or all of the program code for operating system or applications are located in the EEPROM or flash ROM. Since Soft masks allow errors to be corrected and programs to be modified quickly and at minimal cost, they are used primarily during testing and in the field trials.

This paper introduces a hardware architecture of IC card for soft masks. We suggest a new down loading scheme for soft-mask ROM connected by an I/O interface. This scheme saves the new IC card development cost and time.

I. 서론

IC카드 시스템의 수행 코드를 chip 제작 시에 변경이 불가능한 ROM에 masking하는 것을 "hard masking"이라 하고 반면 이 코드를 chip 제작 후에 EEPROM 또는 Flash ROM과 같은 rewrite가 가능한 메모리에 입력하는 것을 "soft masking이라 한다. IC카드는 최소형 컴퓨터로 하나의 chip에 전체 hardware 기능과 OS가 통합된 시스템으로서 개발 단계에서 hardware와 OS를 동시에 개발하여 단일 chip으로 만드는 것은 서로 간의 기능 검증을 수행하기에 상당한 어려움이 있다.

OS를 변경이 불가능한 ROM hard mask로 제작하기 위해서는 IC카드 chip의 Hardware뿐만 아니라 OS 코드들이 충분한 test가 이루어져서 기능상의 오류가 없다는 것이 보장되어야 한다. 만약 서로 간의 충분한 Test를 수행하지 않고 OS를 ROM code화 하여 IC카드를 제작한 후 오류가 발생하는 경우 hardware와 OS 중 어느 곳에서 오류가 발생한 것인지를 확인하는 것은 사실상 불가능하다. 그리고 이 경우는 IC카드 전체의 fail을 의미한다. 반대로 soft mask의 경우는 코드상의 오류에 대하여 정정이 가능하므로 그 만큼의 위험요인과 손실을 줄일 수 있는 장점을 가진다. 그러나 EEPROM이나 flash ROM의 경우 ROM보다 많은 면적과 비용을 필요로 하는 단점을 가진다.

본 논문은 IC카드의 이러한 개발상의 어려움을 해결하기 위해 사용할 수 있는 "soft masking" 구현에 관한 것으로 이를 위해 필요한 IC카드의 hardware구조 및 방법을 제시하였으며 IC카드의 접촉 인터페이스 모듈을 이용한 down loading 방법을 구현하였다.

II. IC카드 시스템

2.1 기본 구조

IC카드 시스템의 기본적인 하드웨어 구조는 그림 1과 같다. IC카드는 IC칩을 내장한 카드로서 메모리와 프로세서를 내장하고 있으므로 저장능력과 연산능력을 가지고 있다. 이전에는 수십에서 수백 바이트 크기의 메모리를 가졌으나, 최근에는 메모리 용량이 수 킬로 바이트에 이를 정도로 메모리 용량이 급속히 늘어나고 있다. IC카드에 내장된 메모리는 카드 OS를 저장하고 있는 ROM, 영구 보존이 필요한 데이터를 저장하기 위한 EEPROM, 그리고 working 메모리로서의 RAM으로 구성된다.

IC카드에 사용되는 프로세스는 암호화 및 복호화 등의 암호연산을 수행하고, 인증, 서명, 프로토콜 처리 등의 작업을 수행한다. 기존의 IC카드에는 8비트 프로세서(8051 계열)를 사용하는 경우가 많지만, 최근에는 다양한 응용 서비스를 수행하고 처리 속도의 고속화를 위해 16비트 혹은 32비트 프로세서를 사용하기도 한다.

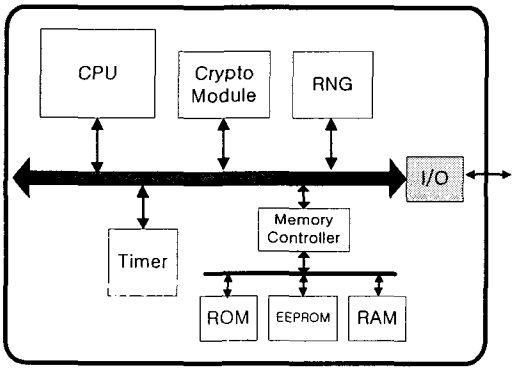

그림 1. IC카드 시스템의 기본적인 하드웨어 구조

2.2 Soft masking의 필요성

종래의 IC카드는 카드 OS를 ROM mask 형태로 카드 chip에 넣는 "hard masking" 방법을 취하고 있다. 따라서 카드 OS를 ROM에 넣기 위해서는 chip과 동시에 카드 OS의 개발이 완료되어야 한다. 그러나 위에서 언급한 바와 같이, 현재의 IC카드의 추세는 다양한 IC카드의 기능을 구현하기 위해 8비트 연산 연산기능을 가지는 CPU에서 32비트 연산이 가능한 CPU로 바뀌고 있으며 이에 따라 카드 OS의 크기가 커지고 기능이 복잡해지고 있다. 또한, 최근에는 Java virtual machine을 탑재한 Java card 등의 개발이 이루어지고 있다. 이러한 IC카드의 OS는 수행 및 기능상의 복잡도로 인해 개발이 상당히 어려울 뿐만 아니라 수행 코드의 크기도 대용량화되고 있다.

카드 OS는 ROM mask로 chip에 넣고 난 후에는 교체 및 변경이 불가능 하므로 카드 OS에 오류가 존재하는 경우, IC카드를 사용할 수 없게 되며 또한 카드 OS의 기능 변경 및 추가가 불가능 하다. 따라서 IC카드 칩을 제작한 후에 카드 OS를 flash 메모리와 같은 비 휘발성 메모리에 입력할 수 있는 "Soft masking" 방식의 IC카드는 그 효용성이 매우 크다고 할 수 있다.

III. IC카드의 soft masking 설계 및 구현

3.1 soft masking을 위한 시스템 구조

"Hard masking"이 가지는 단점을 극복하기 위해서 기존의 IC카드와는 다른 hardware 구조를 제안하였다. 즉, IC카드의 전원이 인가되었을 때 ROM과 플래시 메모리 중에 IC카드를 제어할 프로그램을 선택하기 위한 별도의 입력 핀을 두어 이 입력 핀의 값에 따라 IC카드의 메모리 구조를 변경하고 활성화시키는 메모리 제어 모듈을 가지는 IC카드를 구현하였으며 그림 2에 그 구조를 나타내었다.

또한 기존의 IC카드가 ROM에 시스템의 전체 OS 코드를 입력하는 것과는 달리 본 논문이 제시하는 IC카드 시스템의 ROM에는 플래시 메모리 및 I/O 모듈 제어만을 위한 수행 코드를 입력한다. 따라서 ROM에 입력된 코드의 사이즈는 10k 바이트 미만으로, 단순 기능만을 제공하므로 코드의 오류 발생은 매우 제한적이다.

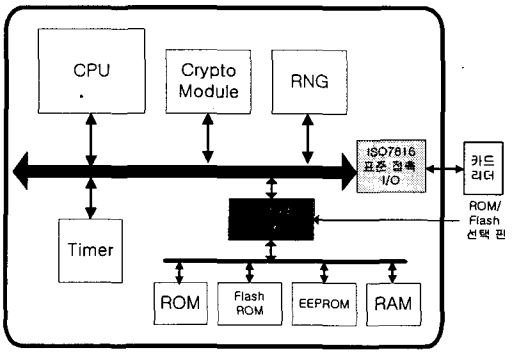

그림 2. "Soft masking"이 가능한 IC카드 시스템의 하드웨어 구조

3.2 작동 원리

그림 2에 표시된 ROM/Flash 선택 핀에 '1'을 입력

한 상태에서 IC카드에 전원을 인가하면 ROM의 기능이 활성화되어 ROM에 입력되어 있는 프로그램이 수행된다. ROM 코드를 이용하여 카드 리더에서 카드로 접촉카드의 표준인 ISO 7816에 명시된 데이터 포맷으로 카드 OS를 입력한 후 플래시 메모리에 저장한다. 이 상태에서의 IC카드 시스템의 메모리 맵은 그림 3과 같다. 그림에서 보는 바와 같이 ROM은 코드 수행을 위해 메모리 맵의 베이스 번지(0x0)에 위치하게 되고 OS 코드가 입력될 flash ROM은 0x10000번지에 위치하여 접촉 I/O를 통해 입력되는 데이터를 저장하는 메모리와 마찬가지로 코드를 저장하게 된다.

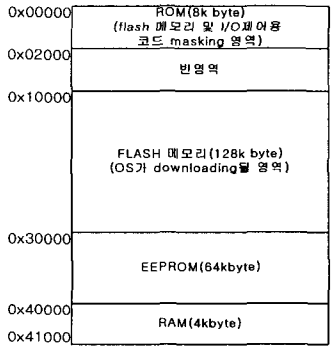

그림 3. ROM 코드 선택 시의 메모리 맵

플래시 메모리에 카드 OS가 입력이 끝난 후에는 ROM/Flash 선택 핀에 '0'을 입력한 상태에서 IC카드의 전원을 인가하면 ROM은 메모리 구성요소에서 제외되고 플래시 메모리가 활성화되어 앞서 플래시 메모리에 입력된 카드 OS가 수행된다. 정확히 말해 ROM/Flash 핀은 "pull down"상태이므로 Flash ROM에 저장된 코드가 수행되는 것이 normal 상태가 된다.

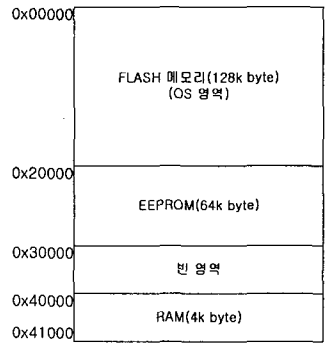

그림 4. Flash ROM에 저장된
카드 OS 수행시의 메모리 맵

이 상태에서의 IC카드 시스템의 메모리 맵은 그림 4와 같다. 그림에서 보는 바와 같이 이 상태에서는 코드 수행을 위한 메모리 맵의 베이스 번지(0x0)에는 flash ROM이 위치하게 되고 ROM은 시스템의 구성요소에서 완전히 제거된다.

3.3 Downloading을 위한 명령어 정의

앞 문단에서 ROM에 masking된 코드를 이용하여 입력되는 OS코드를 flash ROM에 입력한다는 것을 언급한 바 있다. 이와 같은 기능을 수행하기 위해서는 flash ROM을 외부의 리더기에서 제어할 수 있는 명령어를 정의하여 이들 명령어들을 해석하여 수행할 수 있는 코드를 ROM에 masking해 두면 된다. Flash ROM에 코드를 입력하기 위해 정의한 명령어들은 다음과 같다.

(1) Erase 명령어

Flash ROM을 Erase하기 위해 리더기에서 보내는 명령어는 다음과 같이 정의한다.

00 FE XX YY 00

첫 번째 바이트는 "00"로 정의하고 두 번째 바이트는 Erase 명령어를 나타내기 위해 "FE"로 정의한다. Erase는 chip erase와 sector erase로 나눌 수 있는데 세 번째 바이트인 XX에 "00"를 입력하면 chip erase가 되고 "01"을 입력하면 sector erase가 된다. sector erase의 경우는 sector의 위치 정보가 필요하므로 네 번째 바이트에 이를 나타낸다. 사용된 flash ROM의 sector 크기가 4k 바이트 이므로 네 번째 바이트는 4k 바이트마다 1씩 증가한다. chip erase의 경우는 네 번째 바이트는 아무 의미를 가지지 않는다. 다섯 번째 바이트는 "00"으로 정의한다.

(2) Program 명령어

Flash ROM에 코드를 입력하기 위해 리더기에서 보내는 명령어는 다음과 같이 정의한다.

00 FD X1 X2 20

첫 번째 바이트는 "00"로 정의하고 두 번째 바이트는 program 명령어를 나타내기 위해 "FD"로 정의한다. 세 번째, 네 번째 바이트의 X1은 상위 바이트, X2는 하위 바이트를 나타내며, 입력된 코드를 저장할 위치를 지정하게 된다. 다섯 번째 바이트에는 Program 명령어 다음에 리더기로부터 입력될 코드의 바이트 갯수

를 나타낸다. 리더기에서 명령어 뒤에 붙여 보낼 수 있는 데이터의 최대 바이트 수는 32바이트 이므로 명령어의 다섯 번째 바이트에는 "20h"으로 정의한다.

세 번째, 네 번째 바이트에서 나타내고 있는 flash ROM의 program 위치 설정에 대해 좀더 자세히 알아보자. 그림 3에 나타난 flash ROM의 base번지가 0x10000번지이므로 실제로 program이 시작되는 offset 번지도 0x10000번지가 된다. 앞서 설명한대로 한번에 입력되는 코드 크기가 32바이트 이므로 X1 X2의 값은 32 바이트에 1씩 증가 한다. 예를 들어 X1 X2의 값이 01 10이면 0x10000(offset) + 0x0110(X1X2) * 0x20(32 바이트) = 0x12200 번지부터 명령어 뒤에 입력되는 코드를 program하게 된다.

3.4 수행 과정

지금까지 설명한 리더기로부터 카드 OS를 입력 받아 flash ROM에 저장하고, 입력이 완료된 카드 OS를 수행하는 방법을 흐름도로 표시하면 그림 5와 같다.

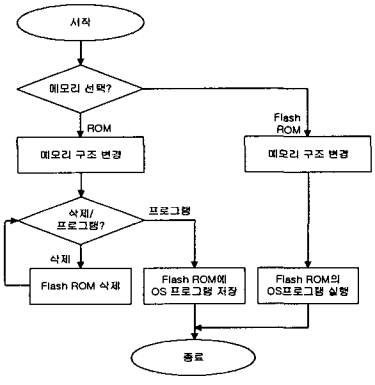

그림 5.수행 과정의 흐름도

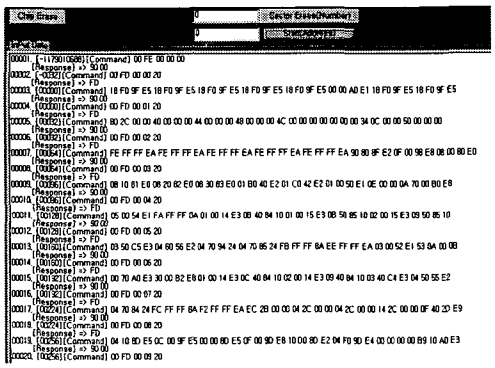

그림 6. 카드 OS의 down loading 과정

그림 6에서는 실제 soft masking 기능이 구현된 칩에 리더기를 통해 위에서 정의한 명령어들을 이용하여 카드 OS를 down loading하는 과정을 보여 주고 있다. 그림에서 보는 바와 같이 첫 번째로 chip erase 명령인 "00 FE 00 00 00"을 보내 flash ROM 전체를 지운다. 카드가 성공적으로 명령어를 수행한 후에는 "90 00"을 응답으로 보내게 된다. 두 번째 명령어부터는 program 명령어를 수행하여 카드OS를 32바이트 씩 flash ROM에 저장한다.

Ⅳ. 결론

본 논문이 제시하는 IC카드는 OS를 chip 제작 이후에 카드 리더를 통해 플래시 메모리에 down loading 시킬 수 있으므로 카드 OS의 개발자는 chip의 제작과 동시에 카드 OS를 ROM에 masking할 필요가 없으며 카드 OS를 필요 시 언제든지 재입력이 가능하게 되어 카드 OS의 오류 수정, 기능의 추가 및 변경이 가능하게 된다.

결국, 카드의 hardware 기능이 향상됨에 따라 카드 OS의 기능이 복잡해지고 다양해지는 현 추세에 따라 큰 용량의 카드 OS를 chip제작과 동시에 ROM masking하는 데서 오는 오류의 수정 및 기능의 변경 불가 등의 위험요소를 제거할 수 있게 되어 IC카드 개발의 효율성을 극대화할 수 있다.

참고문헌

[1] W. Rankl, W. Effing, "Smart Card Handbook", John Wiley & Sons, 2000.

[2] 김영진 외, "Java Card Platform을 내장한 Smart Card의 구현", 정보과학회지, vol. 19, no. 8, pp.34-43, 2001.

[3] 김호원 외, "차세대 IC카드 기술", 한국통신학회지, vol. 17, no. 3, pp.74-83, 2000.