情報保護學會論文誌
第15卷 第5號, 2005. 10

시차와 전력 분석을 이용한 새로운 스마트카드 트랩도어 검출방법*

이 정 엽,[1†] 전 은 아[1], 정 석 원[2‡]
[1]고려대학교, [2]국립목포대학교

A New Method for Detecting Trapdoors in Smart Cards with Timing and Power Analysis*

Jung Youp Lee,[1†] Eun-A Jun,[1] Seok Won Jung[2‡]
[1]Korea University, [2]Mokpo National University

요 약

현재의 스마트카드는, 경제적인 이유 때문에 보안 문제가 발생할 수 있음에도 불구하고, 카드를 재사용할 수 있는 초기화 명령어와 패치 코드를 추가할 수 있는 패치 명령어가 들어있는 경우가 많다. 이러한 트랩도어 명령어들은 세련된 기법으로 만들어지고 일반에게 공개되지 않기 때문에 기존의 테스트 소프트웨어로 검출하는 것은 매우 어렵다. 지금까지 알려진 가장 효율적인 방법은 ITSEC에서와 같이 개발과정을 통제하고 소스 코드를 분석하는 것이다. 그러나 이러한 방법은 오랜 시간을 요구하고 비용이 많이 든다. 이에 본 논문에서는 시차 분석과 전력 분석을 이용하여 스마트카드의 트랩도어를 검출할 수 있는 새로운 방법을 제안한다. 그리고 실험에 의해 제안한 방법이 효율적이며 실질적인 접근 방법임을 보여준다.

ABSTRACT

For economic reasons, even though there are some security problems, the commands of re-initializing and writing patch code are widely used in smart cards. The current software tester has difficulty in detecting these trapdoor commands because trapdoors are not published and programmed sophisticatedly. Up to now the effective way to detect them is to completely reveal and analyze the entire code of the COS with applications such as the ITSEC. It is, however, a very time-consuming and expensive processes. We propose the new detecting approach of trapdoors in smart cards using timing and power analysis. With our experiments, this paper shows that the proposed approach is more practical than the current methods.

Keywords : *Smart Cards, COS, Trapdoor, Timing Analysis, Simple Power Analysis*

Ⅰ. 서 론

스마트카드는 비밀정보를 안전하게 저장할 수 있고 암호 연산기능을 가지고 있어 뛰어난 보안성을 제공한다. 또한 일반 신용카드처럼 다루기 편리하여, 기존의 은행카드와 신용카드로 사용되던 마그네틱 카드가 스마트카드로 빠르게 대체되고 있다. 이처럼

접수일 : 2005년 4월 30일 ; 채택일 : 2005년 8월 19일
* 본 연구는 정보통신부 대학 IT연구센터 육성·지원 사업의 연구결과로 수행 되었습니다.
† 주저자, sonofman@smartgate.co.kr
‡ 교신저자, jsw@mokpo.ac.kr

스마트카드가 주로 금융거래에 사용되기 때문에, 스마트카드를 사용하는 전자지불 시스템의 서비스 제공자는 스마트카드가 시스템에서 정상적인 금융거래를 수행하는 지, 알려진 공격으로부터 안전한 지, 내부에 고의로 넣어놓은 트랩도어가 없는 지에 대해서 검증받기를 원한다.

서비스 제공자는 스마트카드에 대한 다양한 평가와 테스트를 통해 스마트카드를 검증할 수 있다. 평가는 주로 개발과정, 개발 문서, 소스 코드 등을 분석하는 정적인 과정이며, 테스트는 주로 실제 동작중인 스마트카드를 분석하는 동적인 과정이다.

대표적인 평가방법으로는 TCSEC,[1] ITSEC,[2] CC[3]가 있다. TCSEC(the Trusted Computer System Evaluation Criteria)는 1985년 미국 국방부에서 소프트웨어의 신뢰성을 평가하기 위한 기준으로 제정한 것이며, ITSEC(the Information Technique System Evaluation Criteria)는 1990년 TCSEC를 기반으로 유럽에서 제정된 평가 기준이다. CC(the Common Criteria)는 1996년 소프트웨어를 평가하기 위한 국제적인 단일 기준으로 제정되었으며, 후에 국제 표준인 ISO 15408로 표준화되었다. 평가의 기본적인 방법은 사전에 정의해 놓은 보안 위협에 대한 대처 방안에 대해서 등급을 부여하는 것이다. 예를 들면, ITSEC는 E1에서 E6까지의 여섯 등급이 있다. 평가 기관은 소프트웨어가 비정형화된 기능 설명 기준 등 최소한의 기준을 만족하면 가장 낮은 등급인 E1을 부여 하고, 전체 소스 코드와 목적 코드에 대한 분석 등 모든 기준을 만족하면 가장 높은 등급인 E6을 부여한다.

스마트카드의 소프트웨어뿐만 아니라 하드웨어에 대해서도 다양한 평가 기준이 적용될 수 있다. 예를 들면 VISA사는 자신들이 사용할 스마트카드 IC (Integrated Circuit)에 대해서 the Chip Hardware Architecture Review[4]라는 기준을 만족하도록 하고 있다. 이 평가는 스마트카드의 COS (Chip Operating System)나 응용프로그램이 원하는 보안 기능을 구현할 수 있도록 스마트카드 IC가 제공해야 하는 기능에 대해서 기준을 제시하고 있다.

스마트카드의 소프트웨어 테스트는 주로 표준문서에 정의되어 있는 입력에 대해서 스마트카드가 정당한 값을 출력하는 지를 조사하는 것이다. 이러한 입출력에 대한 기능 테스트뿐만 아니라 연산과정에 대한 테스트도 이루어진다. VISA사는 자신들이 사용하는 카드에 대한 인증을 수행할 때, 스마트카드의 암호 연산 과정이 알려진 공격에 대해서 안전한 지 리스크 테스트 과정을 수행한다.

스마트카드의 하드웨어에 대한 테스트는 스마트카드 IC 칩 제조사와 카드 몸체 제조사에 의해서 각 공정마다 수행된다. ATR(Answer To Reset) 테스트나 EEPROM read/write 테스트를 통해서 IC 칩이 모듈에 장착될 때 또는 카드에 고정시키기 위해서 열을 가할 때 물리적인 손상을 입었는지 검사한다.

이러한 다양한 평가방법과 테스트방법에 존재함에도 불구하고, 실제로 사용되고 있는 스마트카드에는 재초기화 명령어와 패치 코드 명령어 같은 트랩도어가 숨겨져 있는 경우가 많다. 재초기화 명령어는 잘못 발급되었거나 사용기간이 만료되어 폐기처분해야 하는 스마트카드를 다시 사용할 수 있도록 EEPROM을 초기화하는 명령어이며, 패치 코드 명령어는 버그가 있거나 스펙이 업그레이드되어 폐기처분해야 하는 스마트카드에 기능을 수정하거나 추가할 수 있도록 패치 코드를 넣을 수 있는 명령어이다. 이러한 명령어들은 경제적으로는 도움이 될지 모르나, 전체 시스템의 안전성에 심각한 위협을 초래할 수 있다. 만약, 실제로 사용되고 있는 스마트카드에 EEPROM을 모두 읽을 수 있는 패치 코드가 추가된다면, 스마트카드의 비밀정보는 손쉽게 노출된다. 특히, 대칭키 암호 알고리즘만을 사용하는 시스템이라면 비밀키의 노출이 시스템 전체의 붕괴를 가져올 수도 있다. 따라서 스마트카드에는 어떠한 형태의 트랩도어도 존재해서는 안 된다.

스마트카드에 존재해서는 안 될 트랩도어들이 실제로는 흔히 존재하는 것은 스마트카드의 트랩도어를 효율적으로 검출할 수 있는 방법이 제공되고 있지 않기 때문이다. 보통 스마트카드가 표준문서에 정의된 명령어들을 정상적으로 수행하고 잘못된 명령어들에 대해서 표준문서에 규정된 에러상태를 응답하면 트랩도어는 존재하지 않는다고 쉽게 생각하지만, 숙련된 개발자라면 이러한 테스트 방법을 피할 수 있는 트랩도어를 쉽게 만들 수 있다. 지금까지 알려진 가장 효율적인 트랩도어 검출 방법은 COS와 응용프로그램의 모든 소스코드를 분석하는 것이다. 그러나 이러한 방법은 오랜 시간을 요구하고 비용이 많이 든다. ITSEC에서 소스코드를 분석하는 가장 하위 등급인 E4를 인증받기 위한 비용은 약 300,000 유로이며, 가장 높은 등급인 E6을 인

증받기 위해서는 수년이 걸리며 수백만 유로를 필요로 한다.[5]

이에 본 논문에서는 시차분석과 전력분석을 이용하는 새로운 트랩도어 검출방법을 제안한다. 기본적인 아이디어는 다음과 같다. 표준문서에 정의되어 있지 않는 명령어들을 스마트카드로 전송하면, 스마트카드의 응답시간과 전력소모는 모두 같다. 하지만, 트랩도어 명령어는 이러한 명령어들과 다른 응답시간과 전력소모를 나타내므로 트랩도어를 검출할 수 있다. 이러한 검출방법은 저렴한 비용으로 구현할 수 있으며 빠른 시간 안에 검출을 완료할 수 있다.

다음 장에서는 실제 스마트카드에서 발생할 수 있는 트랩도어의 형태를 분류해 보며, 3장에서는 트랩도어를 검출하기 위한 현재의 테스트 방법에 대해서 간단히 소개한다. 4장에서는 시차분석과 전력분석을 간단히 소개하며, 5장에서는 시차분석과 전력분석을 이용하여 트랩도어를 검출하는 방법을 제시하고 실험 결과를 보인다.

II. 스마트카드 트랩도어의 유형

스마트카드의 COS는 스마트카드 리더기로부터 명령어를 수신하여, 해당 응용프로그램의 기능을 수행하고, 응답을 송신하는 역할을 한다. 여기서 송수신을 위한 데이터 포맷을 APDU(Application Protocol Data Unit)라고 하며, APDU는 명령어와 응답으로 구성된다.[6] 스마트카드의 트랩도어 또한 APDU 포맷에 맞추어 송수신해야 하므로, APDU 포맷에 대한 분석으로 스마트카드 트랩도어의 형태를 분류해 볼 수 있다.

먼저 APDU 명령어는 그림 1과 같이 명령어 헤더와 몸체로 구성되어 있다. 명령어의 헤더 부분은 클래스 바이트(CLA), 인스트럭션 바이트(INS), 두 개의 파라미터 바이트(P1, P2)로 구성되어 있으며, 몸체 부분은 옵션으로 입력 데이터의 길이 필드(Lc), 입력 데이터 필드(Data field), 출력 데이터의 길이 기대값 필드(Le)로 구성될 수 있다. APDU 응답은 그림 2와 같이 몸체와 트레일러로 구성되어 있으며, 몸체는 선택사항으로 출력 데이터 필드(Data field)가 있을 수 있고, 트레일러는 두 바이트의 상태워드 SW1, SW2로 구성되어 있다.

이러한 APDU 포맷을 고려한다면, 스마트카드의 트랩도어는 크게 세 가지로 분류할 수 있다. 먼저 표준문서나 매뉴얼과 같이 공개된 문서에 정의되어 있는 APDU 명령어에 포함된 트랩도어가 존재할 수 있다. 이러한 트랩도어는 공개된 문서에 정의되어 있는 APDU 명령어의 기능을 수행함과 동시에 숨겨진 기능도 수행한다. 다음으로 공개된 문서에 정의되어 있지 않은 APDU 명령어를 트랩도어로 사용할 수 있다. 이러한 트랩도어는 실제로 스마트카드에서 많이 사용되고 있는 형태이다. 마지막으로 APDU 명령어들을 특정한 순서대로 사용할 경우 마지막 APDU 명령어를 트랩도어로 사용할 수 있다. 본 논문에서는 트랩도어 검출방법을 소개하기에 앞서 트랩도어의 유형을 다음과 같이 정의한다.

정의 1. 은닉 명령어

공개된 문서에 정의되어 있고 스마트카드에 구현되어 있는 APDU 명령어로서, 공개된 문서에 정의되어 있는 기능과 하나 이상의 숨겨진 기능을 동시에 수행하는 APDU 명령어를 은닉명령어로 정의하자.

대부분의 스마트카드 COS는 난수를 응답하는 GET CHALLENGE 명령어를 포함하고 있다.[7] 이 명령어는 은닉 명령어의 좋은 예제가 될 수 있다. 만약 COS가 16 바이트의 난수를 생성하여 응답한다면, 상위 8 바이트는 실제 난수를 발생시키고, 하위 8 바이트는 상위 8 바이트의 난수와 EEPROM 데이터를 XOR 연산하여 16 바이트 응답으로 전송할 수 있다. 이러한 트랩도어가 스마트카드 COS에 구현되어 있다면 외부 프로그램을 이용하여 비밀정보를 포함한 EEPROM의 모든 데이터를 읽어올 수 있다.

정의 2. 트랩도어 명령어

공개된 문서에 정의되어 있지 않고 스마트카드에 구현되어 있는 APDU 명령어를 트랩도어 명령어로 정의하자.

Header	Body
CLA INS P1 P2	[Lc field] [Data field] [Le field]

그림 1. APDU 명령어 구조

Body	Trailer
[Data field]	SW1 SW2

그림 2. APDU 응답 구조

트랩도어 명령어는 다른 정상적인 APDU 명령어처럼 취급할 수 있어서 스마트카드에서 자주 사용되고 있는 트랩도어이다. APDU 명령어는 CLA, INS, P1-P2, Lc, Data, Le로 구성되어 있으므로, 트랩도어 명령어는 CLA 트랩도어 명령어, INS 트랩도어 명령어, P1-P2 트랩도어 명령어 등으로 세분화할 수 있다. 실제로 스마트카드에서는 INS 트랩도어 명령어가 많이 사용된다.

숙련된 개발자는 트랩도어 명령어를 수행하기 전에 하나 이상의 특정 APDU 명령어를 수행하도록 하여, 트랩도어 명령어를 실행할 수 있는 권한을 제한한다. 예를 들면, 패치 명령어를 수행하기 전에 반드시 EXTERNAL AUTHENTICATE 명령어를 정상적으로 수행하도록 한다. 또한 기존의 테스트 소프트웨어가 검출할 수 없도록 트랩도어 명령어에 대한 응답 상태워드로 에러코드 '6D00'(unknown instruction)을 사용한다.

정의 3. 트랩도어 시퀀스

공개된 문서에 정의되어 있고 스마트카드에 구현되어 있는 일련의 APDU 명령어들로서, 사전에 정의된 순서대로 실행되면 마지막 APDU 명령어가 트랩도어로 동작하는 일련의 APDU 명령어들을 트랩도어 시퀀스 명령어로 정의하자.

트랩도어 시퀀스를 GET RESPONSE 명령어, PUT DATA 명령어, GET CHALLENGE 명령어로 가정한다면, GET CHALLENGE 명령어는 일반적인 상태에서는 스마트카드에서 생성한 난수를 응답하지만, GET RESPONSE 명령어와 PUT DATA 명령어를 순서대로 실행한 후에는 EEPROM의 데이터를 읽어 응답하는 트랩도어로 사용될 수 있다. 트랩도어 시퀀스는 실제 사용하는 APDU 명령어들의 조합과 충돌이 일어나지 않도록 정의하여 사용한다.

III. 트랩도어를 검출하기 위한 현재의 테스트 방법

스마트카드의 트랩도어를 검출하기 위해서 사용되는 현재의 테스트 방법은 기능분석이다. 그림 3은 공개되지 않은 APDU 명령어를 검출하는 순서도이다. APDU 명령어의 CLA를 '00'에서 'FF'로 변화시키면서 스마트카드로 전송한다. '6E00'(invalid class) 외의 상태워드를 응답받으면 유효한 CLA 바이트가 결정된다. 그러면 결정된 CLA와 '00'에서 'FF'까지의 INS를 다시 스마트카드로 전송한다. 지원되지 않은 INS는 '6D00'(unknown instruction)이라는 상태워드를 응답할 것이고 지원하는 INS는 다른 상태워드를 응답할 것이다. 이와 유사한 방식으로 사용 가능한 P1-P2 파라미터도 결정할 수 있다. 이러한 방법으로 몇 분 안에 스마트카드가 어떤 명령어를 지원하는지를 알 수 있다. 따라서 공개된 문서에 정의되어 있지 않은 APDU 명령어를 지원한다면 이를 트랩도어로 판단할 수 있다. 이러한 단순 검색 알고리즘이 가능한 이유는 실질적으로 대부분의 스마트카드에서 사용되는 명령어 해독기가 CLA 바이트부터 검색을 시작하여 그 이후의 바이트로 작업을 수행하기 때문이다.

그러나 숙련된 프로그램 개발자는 이러한 테스트에 검출되지 않는 트랩도어를 쉽게 만들 수 있다. 만약 개발자가 트랩도어 명령어에 대해서 항상 '6D00'으로 응답한다면, 이 테스트로는 검출되지 않는다. 또한 이 테스트는 실질적으로 은닉 명령어와 트랩도어 시퀀스도 검출할 수 없다.

IV. 사이드채널 분석

앞장에서 현재의 테스트 방법은 스마트카드의 트랩도어를 검출하기에는 불충분하다는 것을 살펴보았다. 시차분석과 전력분석 같은 사이드채널 분석은 스마트카드에 구현된 암호 알고리즘을 공격하는 강력한 방법으로 알려져 있다. 본 논문에서는 이러한 스마트카드에 대한 공격 방법을 스마트카드의 트랩도어를 검출하는 방법으로 응용하였다. 이 장에서는 시차분석과 전력분석을 간략히 소개한다.

4.1 시차분석

시차분석[10]은 암호 알고리즘 연산에 있어 소요된 수행시간을 측정하여 비밀 정보를 알아내는 분석방법으로 Kocher에 의해 소개 되었다. 공격자가 수행시간을 정확히 측정할 수 있다면 암호 토큰, 암호 시스템 기반의 네트워크 및 그 밖의 시스템에 대하여도 시차공격을 수행할 수 있다. 주로 RSA 등의 공개키 암호 연산이나 DSA와 같은 디지털 서명의 연산에 대한 공격에 많이 적용되고 있다. 이에 대한 알고리즘적인 대응방법으로 RSA 연산시 비밀키 d에 난수를 마스킹하여 연산 시간 분포를 랜덤하게

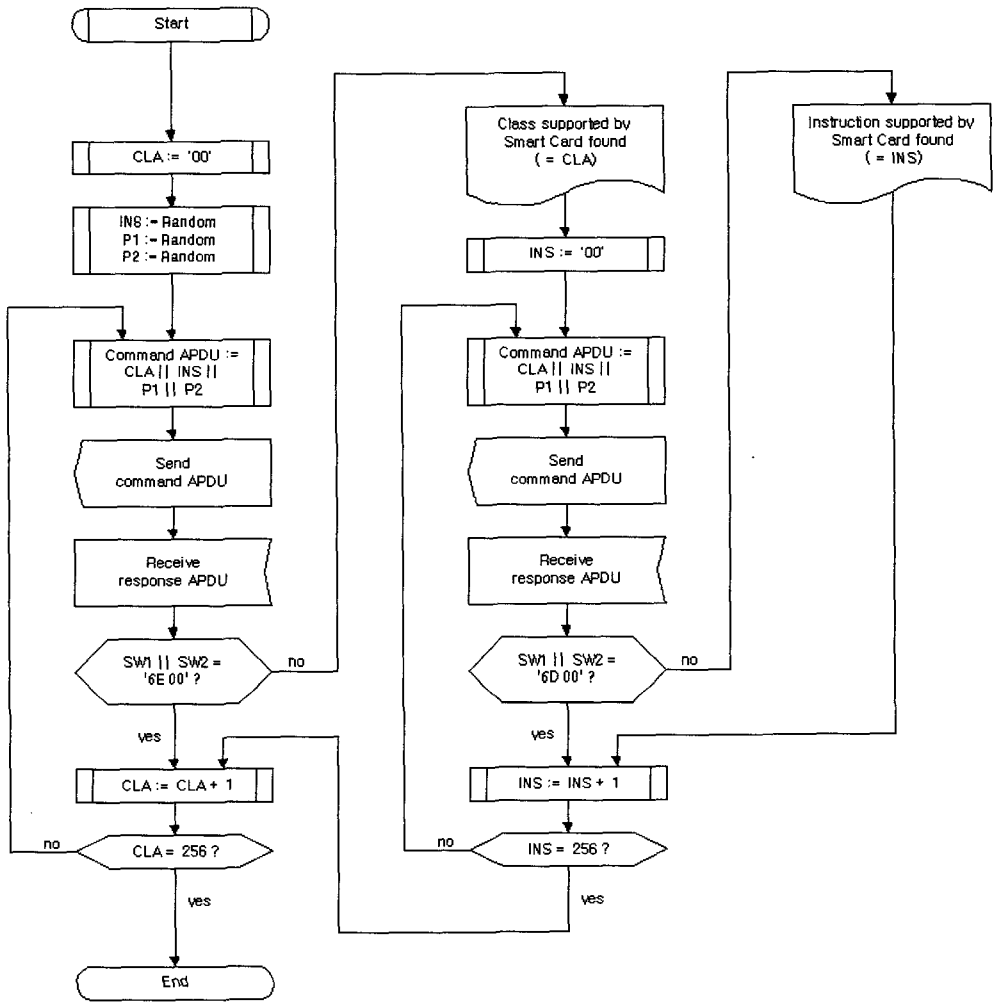


그림 3. 기능 분석에 의한 CLA, INS 검출

나오도록 하거나, 물리적인 대응방법으로 모듈러 제곱 연산만 수행하는 di에 대하여 더미 모듈러 곱셈을 수행 할 수 있는 기능을 하드웨어적으로 제공하여 모듈러 멱승 시간을 항상 일정하게 걸리도록 하는 방법이 있다.

4.2 전력분석

전력분석[11]은 암호 알고리즘이 수행되는 동안의 소비전력을 분석하여 비밀키에 대한 정보를 얻어내는 방법이다. 실제로 구현된 암호 시스템에 대한 공격방법으로 스마트카드와 같은 암호장치에서 서명이나 암·복호화를 수행할 때 논리 게이트의 상태를 관찰하여 입·출력값과 관련되는 누출정보를 얻어내는 방법이다. Kocher가 최초로 DES에 적용하였으며, 현재는 이러한 공격을 견디면서 효율성도 떨어지지 않는 알고리즘들이 제안되고 있다. 전력 분석방법은 크게 SPA(Simple Power Analysis)와 DPA(Differential Power Analysis)로 구분된다.

SPA는 스마트카드에서 연산되는 암호 연산의 소비전력을 직접 관찰하여 카드 내부에 저장되어 있는 비밀키와 수행되는 동안의 명령어에 대한 정보까지 추출이 가능한 공격방법이다. 이에 대한 대응방안으로는 암호 알고리즘 구현 시 조건문의 적절한 사용, 암호연산 시 소비전력 감소, 전력 노이즈 발생 등이 있다.

DPA는 전력소모의 미세한 차이를 통계적인 분석

방법을 사용하여 비밀정보를 분리해 내는 공격방법이다. 이의 알고리즘적인 대응방법으로 DES 연산 과정에서 S-box의 중간 결과 값을 서로 상이하게 나오도록 S-box 연산을 랜덤화시켜서 해당 전력소모량과의 상관관계를 줄이는 방법 등을 사용한다. 물리적인 대응방법으로는 하드웨어 DES를 두 개 사용하여 하나는 정상적인 연산을 다른 하나는 더비 연산을 수행하여 중간 계산 값과 해당 전력 소모량 사이의 상관관계를 줄이는 방법 등이 있다.

V. 스마트카드에서의 트랩도어 검출

대부분의 스마트카드는 문자기반의 비동기식 반이중 전송방식인 T=0 프로토콜을 지원한다. T=0 프로토콜 상에서는 COS가 수신된 APDU 명령어 헤더의 INS를 먼저 분석하여 APDU 명령어 몸체를 수신해야할지 또는 명령어를 수행하고 응답을 송신해야할지 결정한다.[6] 이를 위해서 COS는 일반적으로 알고리즘 1의 명령어 처리 단계를 따른다.

알고리즘 1. APDU 명령 처리 단계

단계1. APDU 명령어 헤더 수신
단계2. INS에 따른 명령어의 해석
단계3. 입력받을 데이터가 존재하는 명령어인 경우 APDU 명령어 몸체 수신
단계4. 보안 메시지 처리
단계5. 표준문서에 따른 입력 명령어 확인
단계6. 명령어 수행
단계7. APDU 응답 전송

만약 스마트카드가 블록기반의 전송방식인 T=1 또는 Type A 또는 Type B 프로토콜[6,8,9]만을 지원한다면, 알고리즘 1의 명령어 처리 단계는 약간 달라진다. 예를 들어 단계 3은 단계 1과 함께 수행될 수 있고, 단계 2와 단계 4는 순서가 바뀌어 수행된다. 블록 전송 프로토콜을 지원하는 대부분의 스마트카드는 T=0 프로토콜도 기본적으로 지원하므로, APDU 명령 처리 단계는 위와 동일하거나 유사한 단계로 구성된다. 본 논문에서는 스마트카드가 T=0 프로토콜 상에서 수행된다고 가정한다.

본 논문에서는 제안한 스마트카드 트랩도어의 검출 방법을 실험하기 위해, 스마트카드 개발 장비를 이용하여 트랩도어가 들어있는 COS를 직접 개발하였다. 실험에 사용된 COS 개발 장비는 삼성전자 스마트카드 IC칩 S3C89V8의 개발 장비인 아이지 시스템의 OPENice i500이며, 스마트카드 단말기는 듀얼아이의 SIR402RE를 사용하였다. 측정 장비는 Tektronix TDS 5052 Digital Phosphor Oscilloscope를 사용하였다. 시차측정은 오실로스코프로 직접 측정하였으며, 전력측정은 스마트카드와 스마트카드 단말기 사이의 Vcc에 50Ω의 저항을 직렬로 연결하여 전압차 측정으로부터 간접적으로 전력을 측정하였다.

5.1 은닉 명령어 검출

은닉 명령어는 정상적인 기능과 숨겨진 기능을 동시에 수행한다. 따라서 알고리즘 1의 단계 6에서 구현되어야 하며, 일반적인 기능 이외의 부가적인 연산을 수행하게 된다. 2장에서 예로든 GET CHALLENGE 은닉 명령어를 살펴보면, 일단 난수를 발생하고 EEPROM 메모리를 읽어온 후 난수와 XOR하여 응답을 한다. 그리고 다음에 읽어올 EEPROM 메모리의 주소를 EEPROM에 기록한다. 대부분의 스마트카드 IC들은 난수 생성을 위한 전용 하드웨어 로직을 IC에 내장하고 있다. 따라서 단순한 기능을 하는 GET CHALLENGE 명령어는 그림 4와 같이 단순한 전력 소모 그래프 모양을 가진다. 반면 부가적인 연산을 수행해야 하는 GET CHALLENGE 은닉 명령어는 그림 5와 같이 부가적인 연산을 수행하고 있음을 보여주고 있다.

은닉 명령어를 검출하기 위해서는 같은 스마트카드 IC를 기반으로 하는 서로 다른 스마트카드 COS들에 대해서 GET CHALLENGE 명령어의 전력 소모 그래프를 다른 카드와 비교하여 부가적인 연산을 수행하는 스마트카드를 트랩도어가 있다고 추정해 볼 수 있다. 실질적으로 은닉 명령어를 찾기 위해서는 스마트카드 IC 특성에 대한 경험이 필요하다.

5.2 트랩도어 명령어 검출

트랩도어 명령어는 2장에서 언급한 것처럼 CLA 트랩도어 명령어, INS 트랩도어 명령어, P1-P2 트랩도어 명령어 등으로 나누어 볼 수 있다. 대표적인 트랩도어인 INS 트랩도어 명령어는 알고리즘 1의 단계 2에서 구현된다. 예제 1에서와 같이 APDU 명령어의 INS 바이트는 switch 문이나 if-else 문에서 비교되어, 해당 함수가 존재하면 해당 함수를

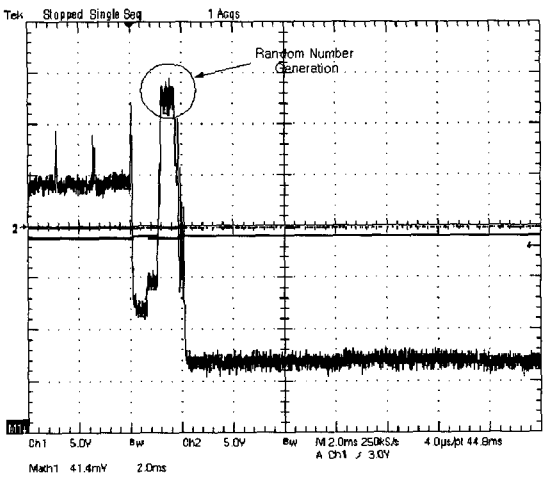


그림 4. 일반적인 'Get Challenge' 명령어의 전력 소모 그래프

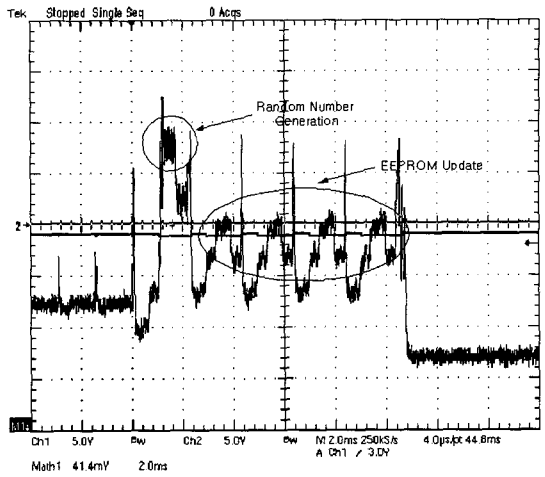


그림 5. 'Get Challenge' 은닉 명령어의 전력 소모 그래프

호출하여 기능을 수행하고 정상적인 상태워드를 응답하며, 그렇지 않으면 '6D00'(unknown instruction) 상태워드를 응답하고 종료한다. 즉, switch 문이나 if-else 문으로 구현된 명령어 해석기에 존재하지 않는 INS 바이트는 항상 일정한 경로를 통해서 수행된다. 따라서 수행시간과 전력소모량은 모두 같을 것이다. 반면 명령어 해석기에 존재하는 INS 바이트들은 서로 다른 경로를 통해서 수행되므로 수행시간과 전력소모량은 서로 다르다.

예제 1. Pseudo-code for command parser and trapdoor function

```
command_parser () {
  switch (apdu.ins) {
    case TRAPDOOR:
         trapdoor();
         break;
    case READ_BINARY:
         read_binary();
         break;
    case WRITE_BINARY:
        if (apdu.lc > 0)
          receive_arr(apdu.body, apdu.lc);
         write_binary();
         break;
     // *** Abbr. ***//
     case GET_CHALLENGE:
         get_challenge(&tApdu);
          break;
     // *** Abbr. ***//
     default:
       send_sw(0x6D00);
       return;
    }
  }
  trapdoor() {
    format_file_system();
    send_sw(0x6D00);
  }
```

INS 트랩도어 명령어를 검출하기 위해서는 공개된 문서에 정의되어 있지 않은 명령어들에 대한 수행시간과 전력소모량을 파악한 후, 다른 명령어들과 다른 수행시간이나 전력소모량을 나타내는 명령어가 존재한다면 이를 INS 트랩도어 명령어로 추정할 수 있다. 그림 6은 명령어 해석기에 존재하지 않는 명령어들에 대한 IO 신호를 측정한 것이며, 그림 7은 INS 트랩도어 명령어에 대한 IO 신호를 측정한 것이다. INS는 0에서 255사이에 값이므로 전체의 INS 트랩도어 명령어를 검색하는 시간은, 하나의 명령어 처리 시간을 30ms라고 가정한다면, 약 8초의 시간이면 충분하다.

만약에 숙련된 개발자가 명령어 해석기의 응답시간을 강제로 지연시켜 트랩도어 명령어와 명령어 해석기에 존재하지 않는 명령어들의 응답시간과 일치시킨다고 할지라도, 스마트카드 IC의 CPU 인스트럭션을 이용하여 소비 전력까지 일치시키는 것은 현실적으로 매우 어렵기 때문에, 전력분석으로 트랩도어 명령어를 검출할 수 있다. 그림 8은 응답 시간이 강제 지연된 명령어 해석기의 전력 소모량을 보여주고 있으며, 그림 9는 INS 트랩도어 명령어의 전력 소모량을 나타내고 있다.

알고리즘 1의 단계 5에서는 파라미터 P1과 P2가

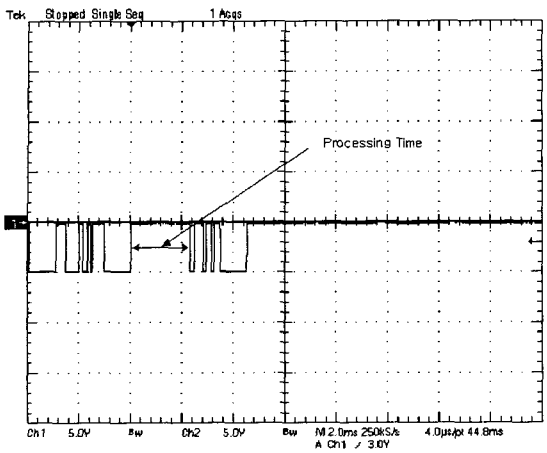


그림 6. 명령어 해석기에 존재하지 않는 명령어들의 IO 신호

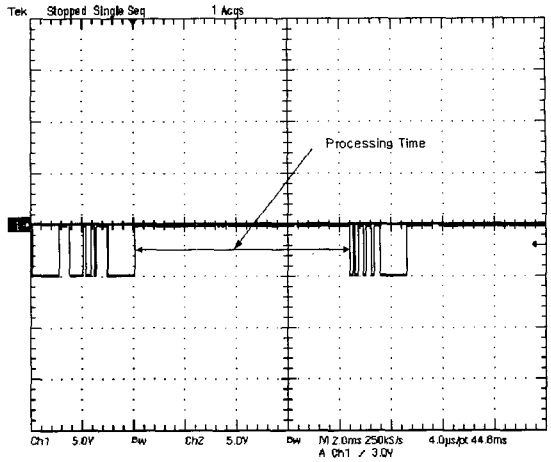


그림 7. INS 트랩도어 명령어의 IO 신호

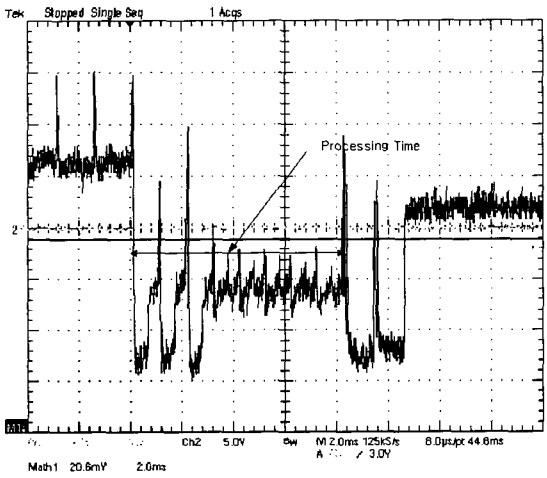


그림 8. 응답시간을 강제 지연시킨 명령어 해석기의 전력 소모 그래프

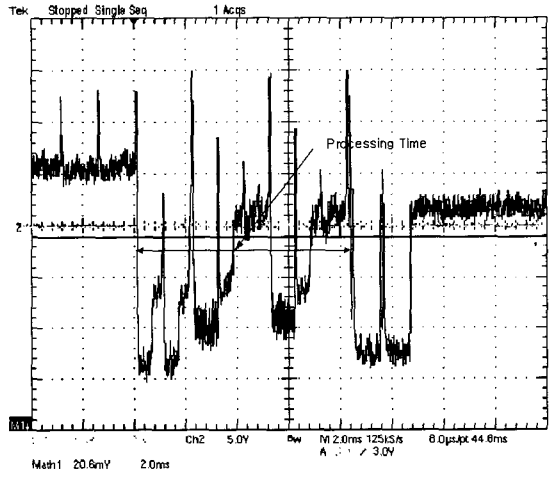


그림 9. INS 트랩도어 명령어의 전력 소모 그래프

표준문서에 맞도록 입력되었는지를 검사한다. 일반적으로 P1을 먼저 검사하고 P2를 나중에 검사한다. 따라서 P1-P2 트랩도어 명령어도 이러한 방식을 따라야 하므로, 만약 P1-P2 트랩도어 명령어가 P1 = 0x37이고 P2 = 0xBF인 경우라면, P1의 값이 0x37 일 때 P2의 값을 검사한다. 이는 응답시간의 미세한 시간차를 유발시킨다. 일치하는 P1 값을 찾는 방법은 INS 트랩도어 명령어를 검색하는 방법을 그대로 적용하면 된다. P2는 P1을 검색한 후 동일한 방법으로 검색하면 된다. 그림 10은 P-P2 트랩도어 명령어에 P1이 0x37 값이 아닌 경우의 응답시간 1.276ms를 보여주고 있으며, 그림 11은 P1이 0x37 값인 경우의 응답시간 1.284ms를 보여주고 있다. 다른 트랩도어 명령어를 검출하는 것은 INS 트랩도어 명령어의 검출 방법과 P1-P2 트랩도어 명령어를 검출하는 방법을 확장한 것이다.

5.3 트랩도어 시퀀스 검출

트랩도어 시퀀스를 구현하기 위해서는 COS 내부에 다음 명령어를 판단할 수 있도록 현재 명령어의 진행 상태를 저장하고 있어야 한다. 2장에서 언급했던 트랩도어 시퀀스를 가정해보자. GET RESPONSE 명령어, PUT DATA 명령어, GET CHALLENGE 명령어 순서로 실행될 때 마지막 명령어가 트랩도어로 동작한다. 현재 상태를 저장하기 위해서

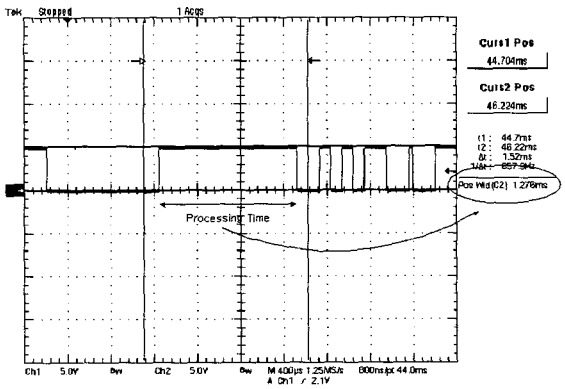


그림 10. P1-P2 트랩도어 명령어가 P1 != 0x37인 경우의 IO 신호

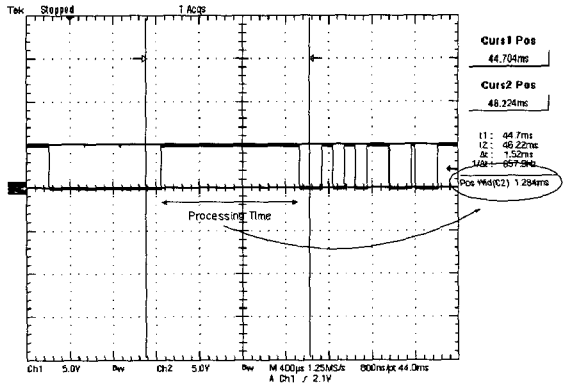


그림 11. P1-P2 트랩도어 명령어가 P1 == 0x37인 경우의 IO 신호

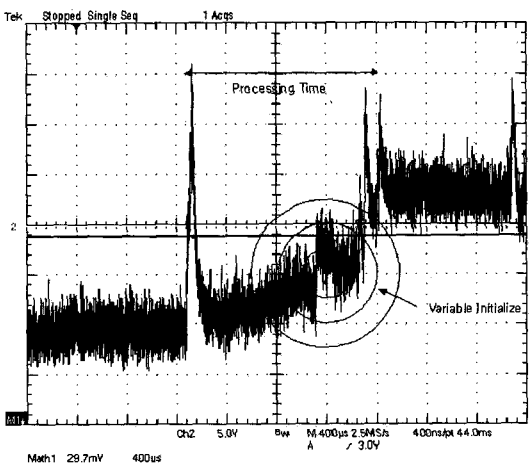


그림 12. 'Get Response'가 아닌 명령어 실행 후 실행되는 'Put Data' 명령어에 대한 전력 소모 그래프

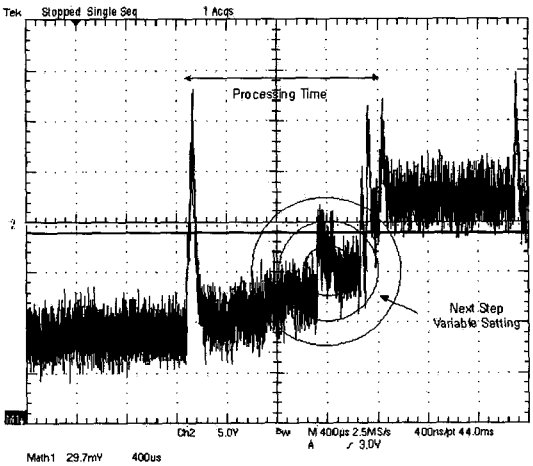


그림 13. 'Get Response' 명령어 실행 후 실행되는 'Put Data' 명령어에 대한 전력 소모 그래프

CUR_STATE라는 변수가 필요하며, CUR_STATE는 초기상태인 S0, GET RESPONSE 명령어가 수행된 상태인 S1, PUT DATA 명령어가 수행된 상태인 S2, GET CHALLENGE 명령어가 수행된 S3의 값을 가진다. CUR_STATE는 현재의 상태에 따라 정의된 다음 명령어가 입력되지 않으면 S0 상태로 바뀐다. PUT DATA 명령어의 관점에서 고려해보면, 이전 명령어가 GET RESPONSE 명령어이면 CUR_STATE의 상태가 S2로 바뀔 것이며, 다른 경우에는 S0로 초기화될 것이다. 그림 12와 13은 이러한 변화에 대한 미세한 전력 소모량의 차이를 보여주고 있다.

트랩도어 시퀀스를 검출하기 위해서는 두 개의 연속적인 명령어를 스마트카드로 전송하는 데, 첫 번째 명령어를 '00'에서 'FF'까지 변화시키면서, 두 번째 명령어도 '00'에서 'FF'까지 변화시킨다. 첫 번째 명령어에 대한 256가지의 두 번째 명령어들의 전력 그래프를 조사하여 CUR_STATE가 다른 값으로 변화하는 명령어들을 찾아 서로의 연결고리를 찾는다. 트랩도어 시퀀스는 공개된 문서에 정의된 명령어들을 사용하므로, 표준문서에 따라 차이가 있지만, 보통 30여개의 명령어들이 존재한다. 그리고 하나의 명령어는 약 10여개 정도의 상태워드를 전송하므로 한 번의 명령어에서 CUR_STATE를 변화시킬 수 있는 가지 수는 300여개로 가정할 수 있다. 두 번의 명령어 수행을 통하여 하나의 연관성을 찾아낼 수 있으므로, 한 번의 명령어 수행에 소요되는 시간을 30ms라고 가정하면, 트랩도어 시퀀스를 검출하기 위해서 약 90분의 시간이 필요하다고 추정할 수 있다.

표 1. 스마트카드의 트랩도어 검출 방법 비교

	기존의 소스코드 분석			기존의 기능 테스트			제안하는 부채널 분석		
	검출여부	검출시간	검출비용	검출여부	검출시간	검출비용	검출여부	검출시간	검출비용
은닉 명령어	○	×	×	×	-	-	△	△	△
트랩도어 명령어	○	×	×	△	○	○	○	○	○
트랩도어 시퀀스	○	×	×	×	-	-	△	○	○

○ : 뛰어남, △ : 보통, × : 떨어짐, - : 지원하지 않음.

Ⅵ. 결 론

실제 현실에서는 COS 개발자들이 경제적인 이유로 스마트카드에 트랩도어를 삽입해 놓는 경우가 흔하지만, 기존의 COS 소스코드 분석방법은 시간과 비용의 문제로, 소프트웨어로 기능 테스트를 수행하는 방법은 이를 무력화시키는 트랩도어를 만들기가 쉬워서, 현실적으로 트랩도어를 효율적으로 검출할 수 있는 방법이 제공되고 있지 않는 상황이다.

본 논문에서는 스마트카드의 트랩도어를 검출하기에 앞서 스마트카드에서의 트랩도어 유형을 은닉 명령어, 트랩도어 명령어, 트랩도어 시퀀스로 정의하고, 이러한 트랩도어들은 기본적인 COS 구조에 의해서 특정 영역에 구현됨을 보였다.

스마트카드의 암호 알고리즘을 공격하는 데 초점을 둔 사이드채널 공격방법은 스마트카드의 트랩도어를 검출하는데 적용될 수 있으며, 시차 분석은 현재 스마트카드의 대표적인 트랩도어인 INS 트랩도어 명령어를 검출하는데 매우 효과적임을 실험을 통해 살펴보았다. 전력 분석은 트랩도어 명령어보다 검출하기 어려운 은닉 명령어나 트랩도어 시퀀스를 검출하는데 사용될 수 있다. 그러므로 본 논문에서 제안한 방법은 표 1에서 정리한 바와 같이 기존의 방법에 비하여 효과적임을 알 수 있다. 또한 제안한 방법은 실제 스마트카드 영역에서 매우 실용적이고 저렴한 방법임을 알 수 있다.

마지막으로 본 논문이 시차 분석과 전력 분석을 이용한 실제적인 스마트카드 내의 트랩도어를 검출할 수 있는 방법들에 대한 연구의 시발점이 되기를 기대한다.

참 고 문 헌

[1] Trusted Computer Systems Evaluation Criteria, US DoD 5200.28-STD, Dec. 1985.

[2] Information Technology Security Evaluation Criteria, Version 1.2, Office for Official Publications of the European Communities, June 1991.

[3] Common Criteria for Information Technology Security Criteria, Version 2.1, Aug. 1999.

[4] VISA Corporation, Chip Card: Testing and Approval Requirements Version 7.0, Industry Services, Dec. 2002.

[5] W. Rankl and W. Effing, "Smart Card Handbook," Third Edition, John Wiley & Sons, Ltd, 2003, pp.244, pp.544-546, pp.579, pp.589.

[6] ISO/IEC 7816-3:1997, Identification cards - Integrated circuit(s) cards with contacts - Part 3: Electronic signals and transmission protocols.

[7] ISO/IEC 7816-4:1995, Identification cards - Integrated circuit(s) cards with contacts - Part 4: Interindustry commands for interchange.

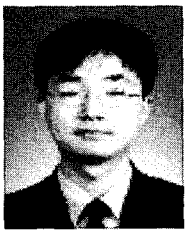
[8] ISO/IEC 14443-3:2001, Identification cards. Contactless integrated circuit(s) cards. Proximity cards. Part 3: Initialization and anticollision.

[9] ISO/IEC 14443-4:2001, Identification cards. Contactless integrated circuit(s) cards. Proximity cards. Part 4: Transmission protocol.

[10] P. Kocher, "Timing Attacks on Implementation of Diffie-Hellman, RSA, DSS, and Other Systems," CRYPTO 1996, LNCS 1109, Springer-Verlag, 1996, pp.104-113.

[11] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," CRYPTO 1999, LNCS 1666, Springer-Verlag, 1999, pp.388-397.

〈著者紹介〉



이 정 엽 (Jung-Youp Lee) 정회원
1998년 2월: 경북대학교 전자공학과 졸업
2000년 2월: 경북대학교 전자공학과 석사
2005년 6월: 고려대학교 정보보호대학원 박사
2002년 1월~현재: (주)스마트 게이트 개발팀장
〈관심분야〉 정보보호, 스마트카드, Side Channel Analysis

전 은 아 (Eun-A Jun) 정회원
1999년 2월: 원광대학교 전기전자정보공학부 졸업
2001년 6월: 원광대학교 전자계산교육대학원 석사
2003년 2월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 정보보호, 스마트카드, Side Channel Analysis

정 석 원(Seok Won Jung) 평생회원
1991년 2월: 고려대학교 수학과 졸업
1993년 2월: 고려대학교 수학과 이학석사
1997년 3월: 고려대학교 수학과 이학박사
2004년 3월~현재: 국립목포대학교 정보보호전공 전임강사
〈관심분야〉 암호 알고리즘 구현, 스마트카드보안, 방송보안