

# PDA에서의 파일 보안 시스템

김희정 · 이주영 · 추영열

동명정보대학교 컴퓨터공학과

## File Security System for Personal Digital Assistants

Hee-jeong Kim, Ju-young Lee, and Young-yeol Choo

Dept. of Computer Engineering, Tongmyong Univ. of Information Technology

### 요약

PDA는 휴대의 편리성과 기능의 향상으로 적용이 확대되면서 통신에서의 보안, 인증, 파일 보안 등 많은 보안상의 요구가 제기되고 있다. 본 연구에서는 PDA 사용자가 중요한 파일을 언제 어디서나 쉽게 보호 할 수 있도록 FSS(File Security System)을 개발하였다. 개발된 FSS 시스템에서는 PDA 내에 저장된 파일 중 보안이 요구되는 중요 파일들에 대해 RC2, RC4 암호 알고리즘을 이용하여 보안 기능을 제공한다.

### 1. 서론

최근 무선 인터넷 사용자의 급증으로 인해 PDA(Personal Digital Assistant)의 발전속도는 훨씬 빨라지고 있다. 이전의 PDA는 PIMS(Personal Information Management System) 기능에 집중되어 있었으나, 현재 출시되고 있는 PDA는 PIMS 기능은 물론 작은 컴퓨터라 불릴 만큼 다양한 컴퓨팅을 소화해 낼 수 있다. 또한 PDA는 PC와의 연결을 통해 동기화를 이루고 있다. PDA의 파일이 PC로 저장될 수 있고, PC의 파일 또한 PDA로 저장될 수 있는 것이다. 근래는 하드웨어의 성능 강화로 휴대용 멀티미디어 기기 등으로 사용이 확대되고 있다.

무선 통신 서비스나 PC와의 동기화, 그리고 모바일 인터넷이 가능해지면서 인증을 통한 메시지 보안, 파일 보안 등 많은 보안 문제가 대두 되고 있다. PDA에 저장된 파일 중 신용카드 번호가 있는 텍스트 문서나 보안이 필요한 이미지 파일 등이 Sync 또는 사용자의 미숙으로 인하여 파일이 유출될 수도

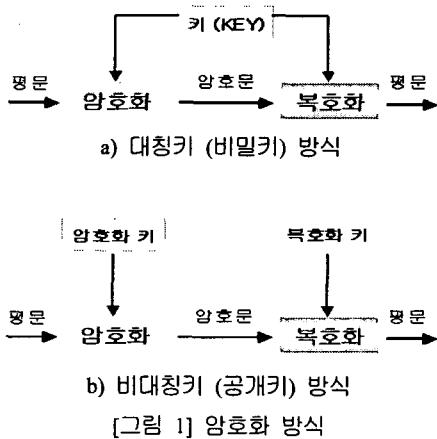
있을 것이다[1].

이와 같은 여러 상황에 대비하여 파일을 보호하기 위해서는 암호화 알고리즘을 이용해야 한다. 암호화 알고리즘은 크게 대칭키(비밀키) 방식과 비대칭키(공개키) 방식으로 구성된다.

대칭키 방식은 [그림 1] a)와 같이 비밀키, 혹은 대칭키라 불리는 하나의 키를 사용하여 원문을 암호화하고 복호화 하는 방식을 말한다. RC2, RC4, DES, Triple DES 등이 이에 속한다[2].

비대칭키 방식은 대칭키 암호화 방식처럼 하나의 비밀키로 암호화하는 것이 아니라 암호화 하는 키(공개키)와 복호화 하는 키(개인키)를 별도로 두는 방식이다. 비대칭키 방식의 대표적인 알고리즘으로 RSA가 있다.[3]

본 논문에서는 데이터 암호화에 적합한 대칭키 방식의 알고리즘 RC2, RC4를 이용하는 파일 보안 프로그램인 File Security System(FSS)을 개발하였다.



본 논문의 구성은 다음과 같다. 제 2 장에서는 File Security System 의 파일 구성도와 설계에 대하여 설명하였고 제 3 장에서는 본 시스템의 개발환경 및 성능 평가에 대하여 기술하였다. 제 4 장의 결론에서는 File Security System 의 향후 연구 방향을 제시하였다.

## 2. File Security System의 구성 및 설계

### 2.1 Fss 의 개념

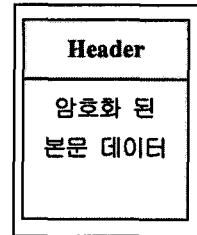
File Security System 이란 PDA에서의 보안을 요구하는 파일을 암호화 알고리즘을 이용하여 보호하는 프로그램이다. 암호 알고리즘은 대칭 키 암호화 방식인 RC2, RC4 알고리즘을 이용한다.

보안을 요구하는 파일을 본 시스템을 통해 암호화 하면 fss 라는 확장명을 가진 암호화 된 파일이 출력으로 생성된다. 사용자는 이 파일을 다시 원문 파일로 복호화 하기 전까지는 이 파일의 내용을 해독할 수 없다.

### 2.2 새로운 파일 형식

일반적으로 특정 형식을 가진 파일은 그 파일 형식(File Format)을 지원하는 전용 프로그램에서만 파일을 액세스 할 수 있다. 또한 확장자 명을 변경하더라도 파일의 형식이 변경되지 않는 한 정상적으로 파일이 읽혀진다. 이는 전용 프로그램이 그 파일의 형식을 정확히 해석하여 브라우저에 파일의 내용을 보여주기 때문이다. 일반적으로 파일 내부에는 고유의 Header나 Tail을 포함하고 있으며 이것으로 파일을

구별하게 된다. 본 시스템에서는 구분을 위해 [그림 2]와 같이 암호화 후 생성된 파일의 구성을 정의 하였다.



[그림 2] Fss 파일의 구성도

암호화 된 파일은 FSS 시스템 및 파일의 암호화 관련 정보를 포함하고 있는 헤더와 원문파일의 평문을 암호화한 본문 데이터로 구성되어 있다.

#### 2.2.1 Header 구조

헤더의 형식은 [표 1]과 같다.

[표 1] Fss Header 구성

Offset	Length	Contents	설명
1	7	FSSv1.0	파일 인식 문자열
9	4	확장자	원문파일의 확장자
15	10	비밀번호	10바이트의 비밀번호
	6	난수	6바이트의 난수

헤더 부분에 포함되는 각 필드의 내용은 다음과 같다.

FSSv1.0

- FSS 형식의 파일임을 인식할 수 있다.

기존 파일 확장명

- 기존의 파일의 속성을 잊지 않기 위해 확장명을 저장하며, 최대 4 바이트로 크기를 고정하였다.

비밀번호

- 사용자가 비밀번호를 최대 10 바이트 까지 입력할 수 있게 한다.

난수

- 6 바이트의 난수를 생성하고 4 바이트 간격으로 헤더에 삽입하여 비밀번호 중복을 방지한다.

헤더 내용은 일반 사용자가 헤더 부분을 해독하

여 본문을 복호화 할 수 없도록 RC4 알고리즘을 이용하여 암호화하고 ANSI 형식으로 파일에 기록한다.

### 2.3 파일 암호화의 설계

원문 파일을 암호화하기 위해서 본 논문에서는 Microsoft 사에서 제공하는 CryptoAPI를 사용하여 대칭 암호화 방식으로 설계하였다.

[그림 3]과 같이 파일을 암호화하기 위해서는 EncryptFile 함수, DecryptFile 함수에 원문 파일명, Fss 파일명, 그리고 세션 키를 생성하기 위한 비밀번호가 인수로 전달된다. 이 함수에서는 CryptoAPI를 사용하기 위해 Key Container의 핸들을 얻어오는 작업을 수행하며 핸들을 올바르게 얻어오면 데이터를 올바르게 암호화 및 복호화 할 수 있다. [3][4]

```

if(m_rOption == ENCRYPT)
{
    if(EncryptFile( PlainTextFileName,
                    FssFileName,
                    Password ))
        AfxMessageBox(_T("암호화 성공"));
    else
        AfxMessageBox(_T("암호화 오류 "));
}

if(m_rOption == DECRYPT)
{
    if(DecryptFile( FssFileName
                    PlainTextFileName,
                    Password ))
        AfxMessageBox(_T("복호화 성공"));
    else
        AfxMessageBox(_T("복호화 오류 "));
}

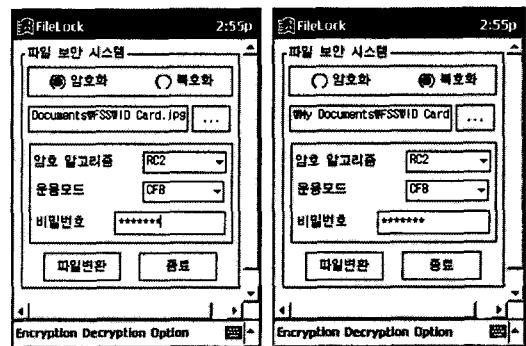
```

[그림 3] 암호화 및 복호화 과정

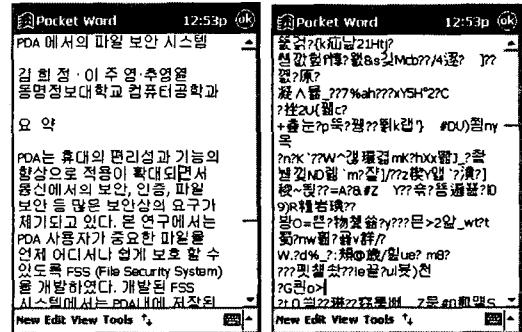
## 3. 구현

PDA환경의 File Security System은 Windows CE 3.0 EVC++을 사용하였으며 PDA는 Microsoft 사의 Pocket PC를 사용하였다. [5][6] 다음 [그림 4]에서는 개발된 FSS의 구현 화면을 보여주고 있으며 [그림 5]는 암호화되기 전의 파일 내용과 본 시스템을 이용

하여 암호화한 후의 파일 본문을 보여준다.



[그림 4] File Security System 구현 화면



[그림 5] 평문과 암호문

암호화 알고리즘을 실제 사용할 경우 각 응용 별 적용을 위해 전자 코드북 (Electronic Codebook, ECB) 모드, 암호 블록 연결 (Cipher Block Chain, CBC) 모드, 암호 피드백 (Cipher Feedback, CFB) 모드, 출력 피드백 (Output Feedback, OFB) 모드의 4 가지 운용 모드를 구현하였다. 시스템의 동작 과정은 다음과 같다.

- 1) Encryption, Decryption 중 하나를 선택한다.
- 2) 암호화 할 경우 보호 대상 파일을, 복호화 할 경우 FSS 파일을 선택한다.
- 3) 암호화 알고리즘은 RC2, RC4 알고리즘 중 하나를 선택한다.
- 4) 비밀번호를 입력한다. 파일을 암호화하거나 복호화 할 때 같은 비밀번호를 입력하여야 올바른 결과값을 얻을 수 있다.

5) 파일변환 버튼을 선택하면 원문파일은 FSS 파일로, FSS 파일은 원문파일로 변환된다. 그러나 복호화 할 경우 암호화 시 입력한 내용과 어느 하나라도 일치 하지 않을 경우 오류메시지를 출력한다.

### 3.1 성능 평가

[표 2]에서는 파일의 크기별로 암호화, 복호화가 시작되는 지점 이후의 속도를 보여준다.

파일 크기가 작을 때는 암호화가 복호화보다 속도가 조금 더 빠르지만, 크기가 커질수록 그 차이는 좁혀진다. 그러나 사용자는 속도 차이를 거의 느낄 수 없고, 반응 시간이 빠르기 때문에 사용에 큰 불편함이 없을 것이다.

[표 2] 파일 크기별 암/복호화 속도

데이터 크기	RC4 알고리즘	
	암호화 속도(Sec)	복호화 속도(Sec)
1 KB	0.00018248	0.00056678
10 KB	0.00064997	0.00111925
100 KB	0.00518815	0.00448560
1,000 KB	0.14676234	0.11563788
10,000 KB	1.36700547	1.289637815

### 3.2 구현의 문제점 및 해결방안

PDA 는 모든 문자를 UNI CODE 로 처리한다. 일반적으로 UNI CODE 기반의 데이터를 파일로 저장하기 위해서는 ANSI CODE 로 변환 후 저장하여야 한다. 그 이유는 UNI CODE 는 2 바이트로 되어있고 ANSI CODE 는 1 바이트와 2 바이트로 되어 있기 때문에 변환 시 데이터의 크기를 알아야 하기 때문이다.

이를 위해 PDA 에서는 UNI CODE 에서 ANSI CODE 로, ANSI CODE 에서 UNI CODE 로 변환해 주는 함수가 정의되어 있다. 아래의 함수를 이용하여 필요할 때 적절한 문자열로 변환하면 된다.[7]

- WideCharToMultiByte
- MultiByteToWideChar

본 논문에서는 사용이 확대되고 있는 PDA 환경에서의 정보 보호를 위한 시스템으로 File Security System 을 개발하였다. 암호화 알고리즘으로는 RC4, RC2 를 구현하였으며 RC2 의 운용 모드로는 각 응용 별로 선택이 가능하도록 전자 코드북, 암호블록 연결 암호 피드백, 출력 피드백 등의 4 가지 모드를 구현하였다. RC4 알고리즘의 파일 크기별로 암호화, 복호화 속도를 측정한 결과 실제 사용 시 커다란 차이가 없음을 알 수 있었다.

### [참고문헌]

- [1] 고재관, 팔아라 실무용 PDA 프로그래밍, 삼각형 프레스, 2002.
- [2] Seungjo Han, Myungshin Oh, "A Study on the Authentication using the Data Compression and Encryption Algorithms," Basic Sciencd and Engineering vol.1 no.1 pp. 969-974, 1997.
- [3] William Stallings, Cryptography and Network Security : Principles and Practice, Prentice Hall, 2002.
- [4] 강선영, Visual C++ 암호화 프로그래밍, 프리렉, 2002.
- [5] [\[5\] http://msdn.microsoft.com/library/](http://msdn.microsoft.com/library/)  
CryptoAPI Reference of MSDN Library
- [6] 여인춘, 김건한, New 알기쉬운 임베디드 비주얼 C++, 정보문화사, 2002.
- [7] Douglas Boling, Programming MICROSOFT WINDOWS CE 2<sup>nd</sup> Edition, Microsoft Press. 2001

## 4. 결론