스테가노그라피를 활용한 정보은닉 응용기법 연구

이 철* · 김용만* · 유승재*

요 약

본 연구에서는 정보은닉을 위한 스테가노그라피에 대한 연구로서 은닉기법에서 비트 플레인 추출방법을 통하여 쉽게 나타나지는 문제점을 보완하였다. 즉, Cover 이미지에 Logo 이미지를 은닉함에 있어 비트 플레인 분산방식 후, Pack 형식을 이용한 코드 빈도 조사와 연속값 조사 등을 통해 압축하여 용량을 줄인 후, Embedding하며 그때 seed 값을 주어 암호화를 높임으로써 쉽게 추출당하지 않을 수 있도록 하였다. 비트 플레인 추출방법은 (0-7)비트 중 어느 한 해당하는 비트에 값을 넣기 때문에 특정 비트에서 값이 보여 지는 문제점이 발생하기 하므로 이를 보완하기 위해 Permutation 후 흩어져 있는 로고 이미지를 한쪽으로 쉬프트 시키는 Permutation 보완작업과 비트 플레인 분산삽입 방식, 팩 형식의 압축 프로그램 등을 적용하였다.

Information Hiding Application Method Using Steganography

Cheol Lee* · Yong-Man Kim* · Seung-Jae Yoo*

ABSTRACT

In this study, we try to make up for the vulnerability in steganography that it is easily revealed the hidden logo image in cover image by bit-plane extraction. For this, we apply some methods, the permutation which shift the scattered pieces of logo image to one side, bit-plane dispersion insertion method and pack-type compressor.

Key words : Steganography, Permutation, Bit-plane Dispersion

접수일 : 2010년 5월 7일; 채택일 : 2010년 6월 18일

* 중부대학교 정보보호학과

1. 서 론

최근 인터넷의 보급 확산과 영상, 오디오 및 동영상 등 대용량 디지털 자료의 가공·처리기술 발전에 편승하여 이들 디지털 콘텐츠의 불법적인 복제나 유통으로 인한 지적재산권 보호문제가 크게 부각되고 있다.

디지털 콘텐츠의 불법 복제와 유통은 저작권자에게 심각한 경제적 손실을 입힐 뿐만 아니라 창작 의욕을 상실시키는 심각한 경제·사회적인 문제이다. 따라서 디지털 콘텐츠의 건전한 거래 질서를 확립하고 창작 활동을 활성화시키기 위해서는 디지털 콘텐츠에 대한 안전하고 견고한 보호 기술 연구와 실용화가 요구되고 있다.

본 연구에서는 정보보호 기술의 중요한 부문으로 부각되고 있는 DRM의 핵심기술인 워터마킹, 핑거프린팅의 기본이 되는 스테가노그라피를 활용한 정보은닉 응용기법에 대해 연구하였다.

인간 시각 시스템을 기반으로 하는 LSB(Least Significant Bits)기법 및 은닉 영상에 적용 가능한 암호기법(DES, Transposition Cipher 등)을 조사하였다. 또한 기존방법의 취약점 해결하기 위해 스테가노그라피 기법과 암호화 기법을 병합하는 방법 연구 및 취약점 분석 을 통해 기존 은닉 영상 추출기법에 강인성을 가지는 암호화 기법 사용을 추구하였다.

2. DRM과 스테가노그라피

2.1 DRM(Digital Rights Management)

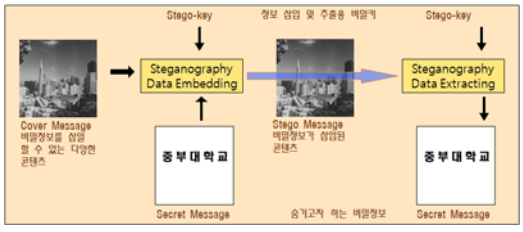
DRM은 콘텐츠 제공자의 권리와 이익을 안전하게 보호하며 불법복제를 막고 사용료 부과와 결제 대행 등 콘텐츠의 생성에서 유통·관리까지를 일괄적으로 지원하는 기술이다. 여기에는 적법한 사용자만 콘텐츠를 사용하고 적절한 요금을 지불하도록 만드는 디지털 저작권 관리기술, 저작권 승인과 집행을 위한 소프트웨어 및 보안기술, 지불·결제기술이 모두 포함된다.

콘텐츠 식별자인 DOI(Digital Object Identifier), 전자상거래에 필요한 데이터를 기록하는 인덱스, 불법복제와 변조방지를 위한 워터마킹 기술을 뒷받침으로 하고 있다. DOI는 디지털 콘텐츠에 부여하는 식별번호이며, 워터마킹은 기밀정보를 디지털 데이터에 숨긴 후 저작권 분쟁이 발생했을 때 디지털 저작권자가 누구인가를 확인할 수 있는 기술이다. 콘텐츠마다 보안인증 시스템을 장착하여 일정한 사용료를 지불하지 않으면 그 콘텐츠를 이용하지 못하게 한다.

DRM은 웹을 통한 유료 콘텐츠의 안전한 배포를 보장하고, 보다 중요한 것은 불법 배포를 방지하기 위해 개발된 서버 소프트웨어의 한 종류이다. DRM 기술은 냅스터와 같은 사용자 간 파일 교환 프로그램들의 광범위한 사용으로 급격히 증가된 상용 제품의 온라인 프라이버시 보호 수단으로 개발되고 있다. 비록 온라인 콘텐츠가 저작권법에 의해 보호 받고는 있다지만, 불법적인 웹 사용을 단속하고 범법 행위자를 잡는 것은 현실적으로 매우 어렵다. DRM 기술은 온라인 밀렵꾼을 범행이 일어난 후 체포하는 마구잡이식 전략보다는, 보다 확실한 문제해결 접근방식으로서 애당초 웹 콘텐츠를 훔치는 것 자체가 불가능하도록 초점을 맞춘다. 많은 회사들이 다양한 접근방법과 기술에 기반을 둔 여러 가지 DRM 제품들을 내놓고 있다. 일반적으로 DRM 제품들은 서버 소프트웨어와 사용자용 플러그인 등, 운영에 필요한 모든 것들이 포함되어 있는 일괄 패키지 형태를 갖추고 있다.

2.2 스테가노그라피(Steganography)

스테가노그래피는 전달하려는 기밀 정보를 이미지 파일이나 MP3파일 등에 암호화해 숨기는 심층 암호 기술이다. 스테가노그래피 기법은 겉보기에나 귀로 듣기에는 암호를 포함했을 때나 그렇지 않을 때나 전혀 차이가 없고, 전송 시에도 일반 파일과 동일하게 보인다.



(그림 1) 스테가노그라피 원리

▌Stego-key에 따른 분류

- Pure steganography : Stego-key 교환이 필요없는 방식
- Secret-key steganography : 대칭 암호와 같은 비밀키를 사용하는 방식
- Public-key steganography : 공개키와 개인키를 사용하는 방식

Cover Message를 삽입할 이미지 선택 후, secret Message를 Stego-Key를 이용하여 embedding 시켜 cover 이미지에 삽입시킨다. 이렇게 삽입시킬 때 핵심은 영상 화질 저하를 최소화하는 등 원본의 외형적(화질, 음질 등) 변화를 최소화 하면서 많은 정보를 삽입하는 것이 핵심이다. 이렇게 삽입된 이미지는 육안으로는 쉽게 내용이 들어있다는 것을 알아 볼 수 없게 되어 비밀 메시지를 숨길 수 있다.

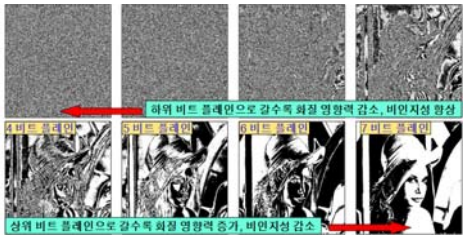
다시 반대로 복호화시에는 암호화 시에 사용하였던 Stego-Key를 이용하여 다시 복호화를 사용하여 추출하면 암호화에 사용하였던 Secret Message를 추출할 수 있다.

3. 응용 스테가노그래피 구현

3.1 비트 플레인(Bit Plane) 삽입

- 영상 화소 값의 특정 비트 위치에 비밀 정보를 적절한 연산방식에 의거 삽입하는 방식

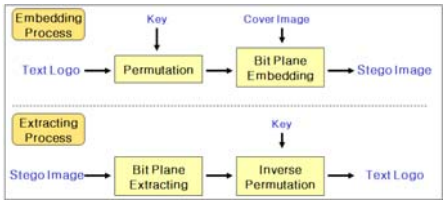
비트 플레인 삽입 방법을 사용 시 하위 비트플레인 쪽으로 갈수록 화질 영향력이 향상 되고, 상위 비트로 갈수록 위의 그림과 같이 비인지성이 감소한다. 따라서 비트 플레인 삽입 방법을 사용할 때에는 0~4비트 플레인 쪽으로 삽입하는 것이 더욱 안전하다.



(그림 2) 비트 플레인 삽입방법

3.2 Permutation 병합

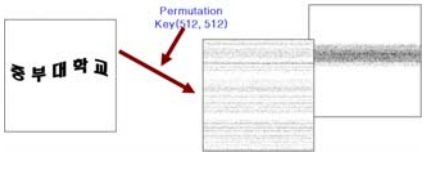
- 비밀정보를 우선 Permutation한 후 비트 플레인 삽입 방법 을 부가하는 방식



(그림 3) Permutation 병합 방법 원리

Permutation 방법은 기존 비트 플레인 삽입 방법을 사용하면, 비트 플레인 추출 방법을 이용하면 로고 이미지가 보이는 문제점이 발생한다.

이 문제점을 해결하기 위하여 Permutation을 한 다음 cover 이미지에 logo 이미지를 삽입하는 방법으로 암호화 방법을 사용하게 되는 것이다.



(그림 4) Logo 이미지 Permutation 실행화면

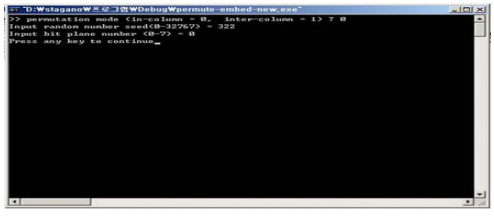
Permutation 방법은 중부대학교라는 로고 이미지를 Permutation을 이용하여 좌우, 또는 상하 좌우로 섞어 버리는 것으로 위의 그림처럼 흑백 TV처럼 값이 나오게 된다.

(그림 5) Permutation에 쓰일 cover image, logo image

(그림 6) Permutation 작업 후 비트 플레인 삽입 결과 화면

Permutation을 추가하여 비트 플레인 삽입을 하였을 경우 비트 플레인 추출방법을 사용하여 특정 비트 값을 뽑아 낸다하여도 위의 그림처럼 비트 플레인 방법은 7비트 플레인에서 Logo 이미지 값이 나와 버리지만, Permutation을 사용하게 되면 밑의 그림처럼 흑백 TV처럼 보이는 듯한 효과가 발생하여 육안으로는 그 내용을 확인할 수 없게 된다.

하지만 Permutation 작업을 하게 된다면, 비트 플레인 추출방법을 사용하여 추출하여도 알아볼 수 없게 좌우, 상하로 섞어 놓지만, Permutation 작업을 하여 비트플레인 추출 방법을 사용하면 추출한 비트 값이 흑백 TV처럼 사진에 무엇인가 있다는 것 정도는 알아 볼 수 있게 된다. 물론 그 내용이 무엇인지 알아볼 수 없을 정도는 되지만, 이렇게 된다면 현재 이사진에 무엇인가 있다는 것을 알 수 있기 때문에 보안상의 취약점이 남아 있게 된다.



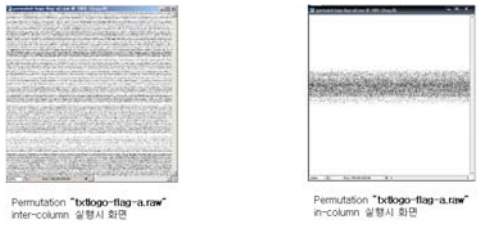
(그림 7) Permutation 프로그램 실행화면

(Permute-embed-new.exe), (Permute-extract-new.exe)

permutation 방식을 이용하여 로고이미지와 커버이미지를 합성시키는 프로그램으로 permutation mode, random number seed번호 설정, bit plane number번호 설정을 할 수 있다.

- Permutation mode : in-column, inter-column = 로고 이미지를 좌우로 섞을 것인가, 상하 좌우로 섞을 것인가 선택 모드
- Input random number seed(0-32767) : 암호화를 위하여 0~32767사이의 seed 번호를 지정 복호화 시에 이용한다.

(그림 8) Permutation 실행 결과



(그림 9) Permutation logo 파일 적용 결과

• Input bit plane number(0-7) : 비트 플레인을 0~7중 어떤 것으로 할 것인지 지정하는 모드

4. 비트 플레인 분산삽입 및 압축

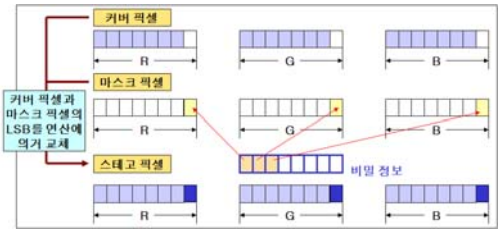
4.1 비트 플레인 분산삽입 방식

비트 플레인 방식이란 현재 LSB 방식은 예를 들어 첫 번째 1을 0비트 플레인에, 두 번째 0도 0비트플레인에, 세 번째 1도 0비트 플레인에 삽입하는 방식으로 모든 로고이미지를 0비트 플레인에 삽입하는 방식이므로 0비트 플레인만 뽑아 낸다면 로고이미지가 나타나 버리는 단점을 갖고 있다.

비트 플레인 분산방식은 기존에 비트 플레인 방식에서는 바이트 단위로 로고 이미지를 삽입하는 방식을 택했으나, 이 방법에서는 1바이트로 전송이 아니라 1비트 단위로 전송을 하게 되어 더욱더 세밀하게 이미지 작업을 하게 된다. 예를 들어 10111010 이라는 8비트 (1byte)를 비트 단위로 각각 비트 플레인 삽입 방식을 하는데, 이때 기존 비트 플레인 방식처럼 한곳에 몰아 놓는 것이 아니라 랜덤하게 전송하게 된다.

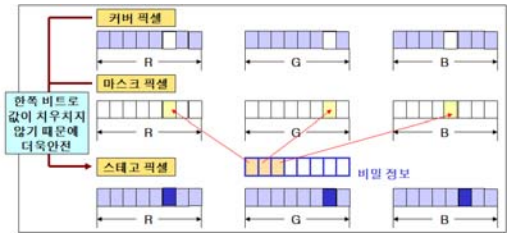
이렇게 하면, 기존 비트 플레인 방식은 0비트 플레인을 뽑아낸다면 이미지가 나와 버리지만, 이 방법은 랜덤하게 들어가기 때문에 한쪽 비트만 뽑아도 제대로 된 값이 나오지 않게 되어 더욱더 보완적으로 강화가 된다.

기존의 LSB 방식은 다음 (그림 10)과 같이 이루어진다.



(그림 10) LSB 방식의 비트 플레인 삽입방식

하지만, 비트 플레인 분산방식은 특정 비트로만 같이 보내는 것이 아니기 때문에 (그림 11)처럼 이루어진다.



(그림 11) 비트 플레인 분산 상입 방식 이해

4.2 Pack 형식

비트 플레인 방식을 하게 된다면 장점도 있지만, 단점이 이미지가 늘어나기 때문에 용량이 늘어난다는 것이다.

1byte 단위로 작업하던 것을 1bit 단위로 작업을 하게 되기 때문에 가로 세로 8배 크기로 늘어나기 때문에 용량 면에서는 효율이 떨어지게 되는 단점이 발생하게 되는 것이다.

그리하여 이 단점을 보완하기 위한 것이 Pack 형식(압축)이다. 이미지 상에서 불필요하게 연속 되는 공간의 이미지를 모아서 압축을 하는 방식이다.

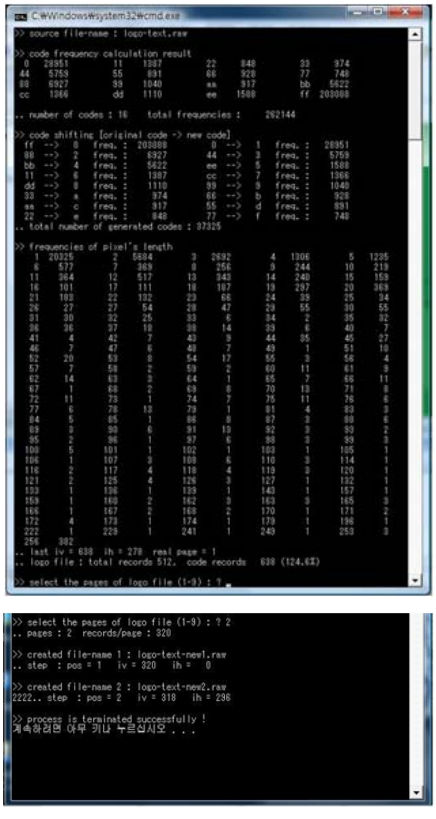
(logo-pack.exe)

Source file-name을 지정하여, 현재 로고이미지의 빈도 조사를 하고, code shifting을 통해서 16진수변환. 그리고 연속적으로 사용되는 픽셀의 값을 찾는다. 이렇게 해서 압축을 하기 위한 총 빈도를 조사 한 후, %값 나온 결과를 토대로 몇 장으로 만들 것인지를 정한다.

- Source file-name : logot-text.raw
  = 소스 파일로 Logo-text.raw 파일을 사용
- Code frequency calculation result
  = 색이 사용된 빈도를 조사
- Number of codes : 16total frequencies : 262144

= 사용된 코드는 16가지이며, 총 빈도수는 262144 이라는 것을 나타냄.

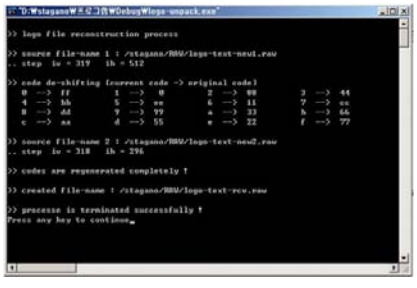
- Code shifting [original code → new code]
  = 코드를 16진수 형식으로 변환시킨다.
- Total number of generated code : 37325
  = 변환된 총 코드는 372325개
- Frequencies of pixel's length
  = 연속적으로 사용된 빈도값을 조사
- Select the pages of logo file(1-9) : ? 2
  = pack을 통하여 압축 후 나온 % 값에 따라 장수를 정한다.
- Created file-name 1 : logo-text-new1.raw
  Created file-name 2 : logo-text-new2.raw
  = 제 2장으로 정했을 경우 생성된 파일들의 이름



(그림 12) Pack 형식 프로그램 실행 화면

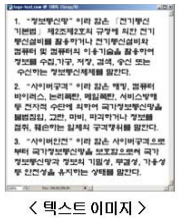
(logo-unpack.exe)

Pack을 통하여 압축된 파일 logo-text-new1.raw, new2.raw 파일을 다시 불러 들여 코드를 오리지널 코드로 변경 후, logo-text-rcv.raw 파일로 복호화한다.



(그림 13) logo-unpack.exe 실행화면

이렇듯, 비트 플레인 분산방식은 암호화하기에 좋으나, 8배로 크기가 늘어난다는 단점이 있었는데, 이런 Pack이라는 압축 방식을 사용하여 그 단점을 해결되었다. 위의 (그림 14)에서 보는바와 같이 왼쪽에 글씨가 많이 쓰여 있어서 압축 효율이 100%가 넘어가기 때문에 두 장으로 나누어 넣으면 오른쪽 그림처럼 나오게 된다. 이런 그림이 제 2장이 되는 것이다. 따라서 압축이라는 암호화를 다시 한 번 하기 때문에 안전성에서 더욱더 효율적이게 된다.



< 텍스트 이미지 >

< 압축 이미지 >

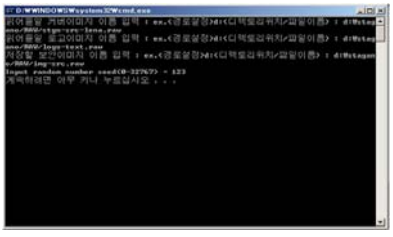
(그림 14) Pack 형식 logo 파일 실행 전, 후 결과

(bit-emd.exe) (bit-ext.exe)

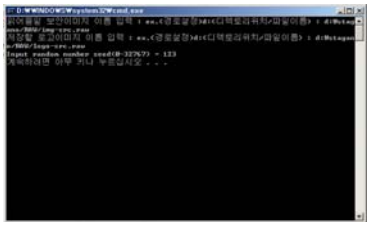
- bit-emd.exe-
  Pack 형식을 이용하여 암호화가 된 Logo 파일을 Cover 이미지와 Embedding시켜 완성된 스

테가노 이미지를 만든다.

- bit-ext.exe-
  복호화시 사용되는 파일 내용은 bit-emd.exe와 동일하다.



(그림 15) 비트 플레인분산방식 embedding bit-emd.exe 실행화면



(그림 16) 비트 플레인분산 방식 embedding bit-ext.exe 실행화면

5. 결 론

본 연구에서는 Cover 이미지에 Logo 이미지를 비트 플레인 분산방식 후, Pack 형식을 이용하여 코드 빈도 조사, 연속값 조사 등을 통하여 압축을 하여 용량을 줄인 후, Embedding하며 그때 seed 값을 주어 암호화를 높임으로써 쉽게 추출당하지 않을 수 있도록 하였다. 즉 스테가노그라피 기법에서 비트 플레인 추출방법을 통하여 쉽게 나타나지는 문제점을 보완하는 방법을 연구하였다.

비트 플레인 추출방법은 (0-7)비트 중 어느 한 해당하는 비트에 값을 넣기 때문에 비트 플레인 추출방법을 사용하게 특정 비트에서 값이 보여 지는 문제점이 발생하기 때문에 이에 대한 보완방법으로 Permutation 후 흩어져 있는 로고 이미지를 한쪽으로 쉬프트 시키는 Permutation보완작업을 추가하였다. 또한 이 과정에서 비트 값을 뽑아내는 과정의 어려움을 극복하기 위해 비트 플레인 분산삽입 방식을 적용하였다. 즉, 기존의 LSB 방식은 해당 비트에만 값을 넣기 때문에 비트 플레인 추출 공격에 취약함을 보이기에 이 비트 플레인 삽입방식을 업그레이드 하여 특정 비트에만 넣는 것이 아니라, %별로 나누어 해당하는 값의 픽셀 값이 나오면 랜덤하게 나누어 넣는 방법을 적용하였다.

그 결과 한 부분의 비트 값만 추출한다하더라 내용은 알아볼 수 없게 되고, 분산되어있기 때문에 확인이 매우 어렵게 만들었다. 그리고 이 과정에서 1byte 단위로 작업 하던 것을 1bit 단위로 바꾸면서 그림의 용량증가의 문제를 팩형식의 압축 프로그램을 이용하여 해결하였다.

참 고 문 헌

[1] An adaptive steganography of optical image using bit-planes and multi-channel characteristics, 2008 (Journal of the Optical Society of Korea).

[2] International conference on Mobile computing and networking, 1999.

[3] Michael A. Caloyannides, “Privacy Protection and Computer Forensic”, Artech House.

[4] 문일남 외 3명, “유사도를 이용한 비트플레인 기반의 스테가노그라피”, 한국해양정보통신학회논문지, 제13권, 제4호, pp. 684-690, 2009.

[5] 김영실, 박성진, “정보은닉을 위한 스테가노그라피와 DRM”, 인터넷 정보학회지, 제4권, 제1호, pp. 30-36, 2003.

[6] 이신주, 정성환, “비트 플레인별 적응적 임계값을 이용한 대용량 스테가노그라피”, 정보처리학회논문지, 제B b11권, 제4호, pp. 395-402, 2004.

이 철
중부대학교 정보보호학과

김 용 만
중부대학교 정보보호학과

유 승 재
1988년 동국대학교 수학과 (이학사)
1990년 동국대학교 수학과 (이학석사)
1998년 동국대학교 수학과 (이학박사)
1997년~현재 중부대학교 정보보호학과 교수