정보학연구 제5권 제4호 2002년 12월(pp.115-127)
The Journal of Information Technology, Vol.5, No.4

GSF(GrayScale File) 출력을 이용한 3-Tire 파일 암호화 알고리즘

김영실* · 김영미** · 김륜옥*** · 백두권****

요 약

본 논문에서는 ciphertext를 적절한 cover 이미지를 사용하지 않고 은닉이 가능하도록 grayscale 형태의 이미지로 표현하는 개선된 파일 암호화 알고리즘을 제안한다. 제안된 파일 암호화 알고리즘은 기존의 스트림 암호화 알고리즘과 블록 알고리즘을 이용하여 2-Tire 암호화를 수행한 후 3-Tire에서는 암호화된 ciphertext의 구조와 형식을 은닉하기 위해 MBE(Modyfied Block Encryption) 알고리즘을 제안하고 적용하였다. 제안된 GSF 출력을 이용한 파일 암호화 알고리즘은 암호화되어 생성된 이미지 파일이 palintext 파일의 종류에 관계없이 거의 비슷한 패턴을 가지므로 파일의 암호화뿐만 아니라 은닉효과까지 기대할 수 있다. 또한 블록 암호화 알고리즘 적용 시 발생할 수 있는 padding 처리를 위해 SELI(Select Insert) padding을 제안하고 적용하였다.

Ⅰ. 서론

오래전부터 암호화(cryptography)기술은 데이터의 저장(stroage)과 전달(transmission)과정에서 발생할 수 있는 외부의 부정적인 요인(도청, 변조, 삽입, 삭제)으로부터 데이터들을 보호하기 위해 사용되었다. 1960년대 이전까지만 해도 암호와 관련된 개념은 국방과 외교 분야의 전유물이었다. 하지만 컴퓨터와 통신시스템의 비약적인 발전으로 인하여 현재에는 민간 상업분야에서도 컴퓨터와 통신시스템의 보안을 위해 암호화기술이 적용되고 있다.[1][2][4] 더불어 암호화된 ciphertext를 안전하게 보호하기 위한 방안으로 정보은닉 기술도 다양한 형태로 연구되어 적용되고 있다. 이중 대표적인 응용기술이 Stegonography이다. Stegonography는 데이터를 다양한 형태의 자료(텍스트, 이미지, 동영상, 음악)에 은닉함으로써 일반 사용자는 숨겨진 데이터를 찾아 내지 못하도록 지원하는 기술이다. 현재 가장 일반적으로 가장 발전된 분야가 이미지를 이용한 Steganography이다. [3][5]

상용화된 대부분의 파일 암호화 프로그램을 살펴보면 파일을 암호화한 후 암호화 된 ciphertext를 텍스트형태의 파일로 생성하기 때문에 plaintext를 알아낼 수 없다. 그러나 화면상으로 쉽게 ciphertext의 내용을 확인할 수 있기 때문에. 누구나 해당 파일이 암호화 된 파일이라는 것을 쉽게 판독할 수 있다. 본 논문에서는 암호화된 ciphertext를 grayscale 형태의 이미지 파일로 구현함으로써 기존의 파일 암호화 프로

* 대림대학 컴퓨터정보과 조교수
** (주) 세스 연구실장
*** 대림대학 컴퓨터정보과 시간강사
**** 고려대학교 정보통신대학 교수

그램이 가지고 있는 문제점을 보안할 수 있는 GSF(GrayScale File)출력을 이용한 3-Tire 파일 암호화 알고리즘을 제안하고 구현한다.

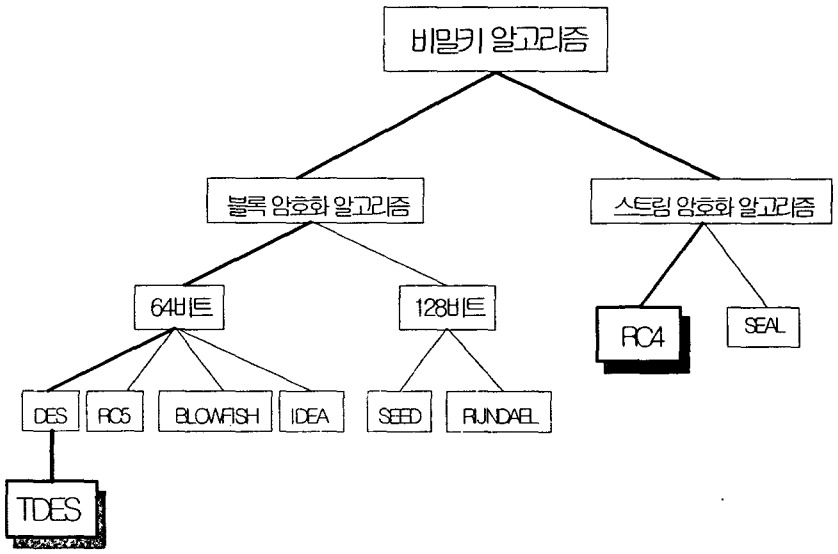
본 논문에서는 2장에서 암호화를 위해 사용되는 대표적인 비밀키 알고리즘의 종류와 운영모드 및 padding 기법 그리고 정보은닉을 위한 Stegonography 기술에 대하여 살펴보고, 3장에서는 암호화된 ciphertext를 이미지형태로 표현해주는 모델을 설계하였다. 4장에서는 설계한 모델을 시스템으로 구현하였으며 5장에서 결론을 기술하였다.

II. 관련연구

비밀 키 알고리즘에는 주어진 plaintext를 일정한 크기의 블럭으로 나누어 암/ 복호화를 수행하는 블록 암호 화 알고리즘과 plaintext 블록으로 나누지 않고 plaintext와 같은 길이의 키 스트림(stream)을 생성하여 평문과 키를 비트단위로 XOR(execlusive OR)하여 암/복호화를 수행하는 스트림 암호화 알고리즘이 있다. [1][2][4][6][7]

2.1 블록 암호 알고리즘

대체(substitution)와 치환(permutation)의 반복에 의해 강력한 암호화 알고리즘이 설계될 수 있다는 Claude Shannon의 이론에 근거한 알고리즘으로 블록으로 나뉘어진 모든 plaintext에 동일한 암호화 알고리즘이 적용된다. 평문의 크기가 블록 크기의 배수가 아닐 경우 블록의 길이를 맞추기 위해 padding 작업이 필요하다. 나뉘는 블록의 길이에 따라 64비트 암호와 128비트 암호로 나뉘어 지며 대표적인 64비트 블록 암호 알고리즘에는 DES와 RC2가 있으며 128비트 알고리즘에는 SEED등이 있다.



(그림 1) 비밀키 알고리즘

DES(Data Encryption Standard)는 평문과 암호문의 길이를 64비트로(실제는 56비트) 구성하여 암호화시키는 알고리즘으로 1977년 공식적으로 채택되어 금융권을 중심으로 폭 넓게 사용되었다. 하지만 하드웨어의 기술수준이 높아짐에 따라 64비트 키의 안전성에 의구심을 가지게 되었고 현재는 기존의 응용이외에는 사용하지 말도록 권고되고 있다. 대신에 IPSec이나 PKIX와 같은 새로운 응용에서는 DES 알고리즘 자체를 변형시키지 않고 안정성을 증대시키기 위한 방법으로 DES를 2개의 서로 다른 키로 3번 암호화하는 Triple DES의 사용을 권고하고 있다.

RC2와 RC5는 RSA사의 Rivert가 설계한 알고리즘으로 RC2는 소프트웨어 보안제품으로 사용되다가 현재는 거의 표준으로 사용되는 알고리즘이며, RC5는 키 길이와 블록의 길이 그리고 라운드 수등을 파라미터로 설정할 수 있는 알고리즘이다.

BLOWFISH는 Bruce Schneier에 의해 설계된 알고리즘으로 충분한 길이의 키를 지원하기 때문에 안전하다고 알려져 있으나 키가 자주 바뀌어야 하는 응용에서는 key setup time이 길어지는 단점을 가지고 있다.

IDEA는 Lai 등이 설계한 16비트 모듈라 곱셈을 비선형 연산으로 사용하는 암호화기법으로 DES를 대체할 수 있는 기법으로 꼽혀왔으며 현재 64비트 블록 암호 기법 중에서 가장 안전한 알고리즘으로 알려져 있다.

SEED는 한국 정보보호센터 주관으로 개발되어 현재 TTA 표준으로 제정절차를 밟고 있는 암호화 알고리즘으로 128비트의 블록과 키를 지원한다. 민수부분용 국내 표준 블록 암호화기법으로 개발되어 현재 다양한 분야에서 응용되고 있다.

RIJNDAEL는 Daemen과 Rijmen에 의해 개발되어 2000년 10월 AES 알고리즘으로 최종 선정된 암호화 알고리즘이다. 다른 알고리즘과는 달리 128비트, 192비트, 256비트의 가변 길이 블록에 128비트, 192비트, 256비트의 가변 길이 키를 사용할 수 있으며 128비트 블록의 128비트, 192비트, 256비트 키를 사용할 경우 각각 10, 12, 14 라운드 사용을 권고하고 있다.

2.1.1 운영모드 및 padding 기법

블록 암호화 알고리즘은 데이터 암/복호화시 여러 개의 블록들을 처리하게 된다. 암/복호화시 적용하려고 하는 블록 암호화 알고리즘을 각각의 블록들 사이에 어떻게 적용할 것인가를 지정해주는 방법이 운영모드이다. 대표적인 운영모드에는 ECB, CBC, OFB, CFB등이 있다. [1][2][4][5][8][11][12]

ECB(Electronic Code Book) 모드는 가장 단순한 운영모드로 나뉘어진 블록 단위로 암호화를 수행한다. 즉 동일한 plaintext 블록에 대해서는 동일한 ciphertext가 생성되며, 암호화된 특정 부분의 내용을 다른 내용으로 변경할 수 있는 문제점을 가지고 있다.

$$\text{암호화} : C_i = E_k(P_i),\ (i=1,\ldots,t) \qquad (2\text{-}1)$$

$$\text{복호화} : P_i = D_k(C_i),\ (i=1,\ldots,t) \qquad (2\text{-}2)$$

CBC(Cipher Block Chaining) 모드는 이전 블록의 ciphertext와 현재의 plaintext 블록을 XOR 연산 수행한 후 그 결과를 암호화하여 ciphertext를 생성하는 운영모드로 맨 처음 블록은 초기화 백터 IV(Initialization Vector)를 사용하여 암호화를 수행하게 된다. 한 블록의 오류는 복호화시 두개의 연속된 plaintext 블럭에 영향을 미친다.

암호화 : $C_i = E_k(C_{i-1} \oplus P_i)$, $(i=1,...,t,\ C_0 = IV)$ (2-3)

복호화 : $P_i = C_{i-1} \oplus D_k(C_i)$, $(i=1,...,t,\ C_0 = IV)$ (2-4)

OFB(Output FeedBack) 모드는 블록 암호를 스트림 암호처럼 처리할 수 있는 운영모드로 암/복호화 과정이 동일하며, 한 블록의 오류는 복호화가 대응되는 하나의 plaintext 블록에만 영향을 미친다.

암호화 : $C_i = P_i \oplus I_i$,$(I_0 = IV,\ I_i = E_k(I_{i-1}))$ (2-5)

복호화 : $P_i = C_i \oplus I_i$,$(I_0 = IV,\ I_i = E_k(I_{i-1}))$ (2-6)

CFB(Cipher FeedBack) 모드는 feedback 의 크기를 1, 8, 16, 32, 64비트 중 선택해서 지정할 수 있으며 작은 단위로 암호화가 되므로 네트워크와 같은 환경에 유용한 운영모드이다. 블록의 크기가 n 비트 이며 feedback size 가 r 비트인 경우, 한 블록의 오류는 n/r+1개의 plaintext 블록에 영향을 미친다.

암호화 : $C_i = P_i \oplus k_i$
(k_i는 $E_k(I_i)$의 상위 r비트), $I_{i+1} = (I_i << r) \oplus (0 \ldots 0 \| C_i)$ (2-7)

복호화 : $P_i = C_i \oplus k_i$, $I_{i+1} = (I_i << r) \oplus (0 \ldots 0 \| C_i)$ (2-8)

특히 ECB 모드와 CBC 모드는 입력 데이터의 길이가 지정된 블록의 배수가 되어야 하기 때문에 마지막 블록의 길이가 64비트 또는 128 비트가 되지 않는다면 임으로 값을 입력하여 길이를 맞춰주어야 한다. 이러한 작업을 padding 이라 한다.

PKCS(Public-Key Cryptography Standards) Padding은 padding 되어야 하는 블럭을 바이트 단위로 추가하여 암호화를 하기 위해 필요한 크기의 블록을 생성하는 방식이다.

OneAndZeroes Padding은 PKCS padding과는 달리 비트 단위로 맨 처음에 1을 추가한 뒤 필요한 만큼 0을 추가하는 방식이다.

CTS(CipherText Stealing) Padding은 plaintext의 마지막 두 블록을 ciphertext의 크기가 plaintext의 크기보다 커지지 않도록 평문의 마지막 블록을 처리하는 방식이다. 이 방식은 입력 데이터가 1블럭 이하이면 적용할 수 없는 단점을 가진다.

Zero Padding은 필요한 길이만큼 0을 추가하는 방식이다.

2.2 스트림 암호 알고리즘

블록암호와는 달리 사용자의 비밀키로부터 생성된 키 스트림을 이용하여 암/복호화를 수행하는 알고리즘이다. 비트단위로 암/복호화가 수행되므로 padding작업이 필요하지 않으며 블록 암호화에 비해 처리 속도가 빠르다. Claude Shannon은 키 스트림을 한번만 사용하는 One-Time Pad 에서의 ciphertext와 plaintext는 통계적으로 독립적이기 때문에 암호분석가에게 무한한 계산 능력과 시간이 주어진다고 할지라도 단지 각각의 평문 비트들을 추축하는 것 이상이 될 수 없다는 의미에서 "깨질 수 없는(unbreakable)" 암호로 칭하고 있다.[1][11] RC4와 SEAL이 대표적인 스트림 암호화 알고리즘이다.

RC4는 1987년 RSA사의 Rivert가 설계한 스트림 암호화 알고리즘으로 가변 키 길이를 지원하며 Nestcape Navigator의 데이터 보호용으로 사용되고 있다.

SEAL는 160비트 키를 사용하는 암호화 알고리즘으로 빠른 소프트웨어 구현을 위해 설계되었기 때문에 소프트웨어적인 대용량의 암/복호화에는 적당하지만 인터넷 응용에서는 거의 사용되지 않는다.

2.3 Steganography

"Covered Writing"의 뜻을 가지는 그리스어로부터 유래된 것으로 통신상의 두 주체인 Sender와 Receiver 사이의 메시지가 제 삼자에게 의심을 받지 않도록 교묘히 숨기는 기법을 의미한다. 즉 Steganography의 목적은 제 삼자가 평범한 일반 메시지 안에 비밀 메시지가 존재한다는 것을 알지 못하도록 숨기는 것이다.

일반적으로 숨겨지는 데이터는 암호화된 Cipher 데이터일 수도 있고 암호화되지 않은 일반 데이터 일수도 있다.

디지털화된 환경에서 이용 가능한 Stegonography는 이미지나 음악, 동영상등에 데이터를 숨기는 기법이다. 항상 noise의 형태로 잉여 데이터가 존재하는 디지털 데이터에 숨기고자 하는 데이터를 noise의 형태로 코딩하여 자연적인 noise의 형태와 구분이 어렵게 데이터를 숨기는 방법이다. 현재 가장 보편화되어 사용되고 있는 Cover 데이터는 이미지이며, 이러한 이미지 데이터에 데이터를 숨기는 가장 간단한 방법은 이미지 데이터 각각의 화소 element에서 가장 적게 사용된 비트에 숨기고자 하는 비밀 메시지의 비트들을 대체 삽입하는 방법이다.

현재 가장 많이 응용되고 연구되는 분야가 바로 디지털 이미지 Steganography이다. 하지만 디지털 이미지 Steganography는 여러 가지 문제점을 가지고 있다.

첫째 사용할 수 있는 Cover이미지의 종류와 크기에 제한적이다. 현재 가장 많이 사용하는 이미지는 JPEG와 BMP이며 비밀 데이터를 숨기기 위해선 비밀 데이터보다 훨씬 큰 Cover 이미지가 필요하기 때문이다.

둘째 숨겨진 비밀 데이터에 의해 실제 Cover 이미지에 왜곡이 발생한다. Cover 이미지를 찾을 수만 있다면 Stego 이미지와 Cover 이미지를 비교하여 시각적인 차이를 쉽게 분석할 수 있다. 이는 8비트 이미지들의 이미지 파레트에서는 색상의 수가 256 컬러로 제한되기 때문에 인접한 이미지 색상들이 비슷하지 않은 이미지에 데이터를 숨기게 되면 확연히 이미지가 구분되기 때문이다. 24비트 BMP를 이용하는 경우에는 왜곡 정도가 심한편이다.

셋째 Cover 이미지가 없다 하더라도 Steganography 알고리즘이 발생시키는 또는 숨겨진 데이터에 대한 식별정보의 삽입으로 발생되는 특이한 유형의 반복적인 형태의 데이터를 분석하게 되면 자료의 구조나 형식을 알 수 없어도 Stego 이미지에 어떤 데이터가 숨겨져 있다는 것을 발견할 수 있다. 이에 Cover 이미지로 Black과 white의 명도 차이를 8비트(256방식)으로 구분하여 표시하는 grayscale 이미지들의 사용을 강조하고 있다. [3][5][7]

III. GSF 출력을 이용한 3-Tire 파일 암호화 알고리즘 설계

기존의 파일 암호화 프로그램은 암호화된 파일을 텍스트 형태의 파일로 생성하기 때문에 누구나 암호화된 파일이라는 것을 알아낼 수 있다. 이러한 단점을 보완하기 위해 응용할 수 있는 것이 Steganography이다. Steganography와

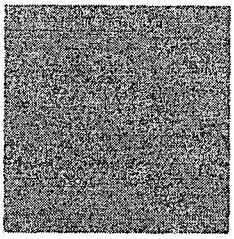
같은 방법은 반드시 숨기고자 하는 데이터 이외에 Cover 데이터가 있어야만 사용할 수 있다. 만약 적절한 cover 데이터가 없는 경우에는 (숨기고자 하는 데이터의 설계 데이터) 이 방법을 사용할 수 없게 된다.

이에 텍스트로 생성된 ciphertext 파일이 적절한 cover 이미지를 선택하지 않아도 은닉이 가능하도록 암호화된 파일을 grayscale 이미지 파일로 만들어 주는 GSF(GrayScale File) 출력을 이용한 3-Tire 파일 암호화 알고리즘을 제안하고 설계한다.

3.1 모델설계

일반적으로 암호화된 ciphertext를 이미지로 표현하게 되면 파일의 종류에 따라 특정한 형태의 패턴이 나타나게 된다. 다음은 다양한 암호화 알고리즘으로 암호화된 ciphertext를 이미지로 표현한 결과이다.

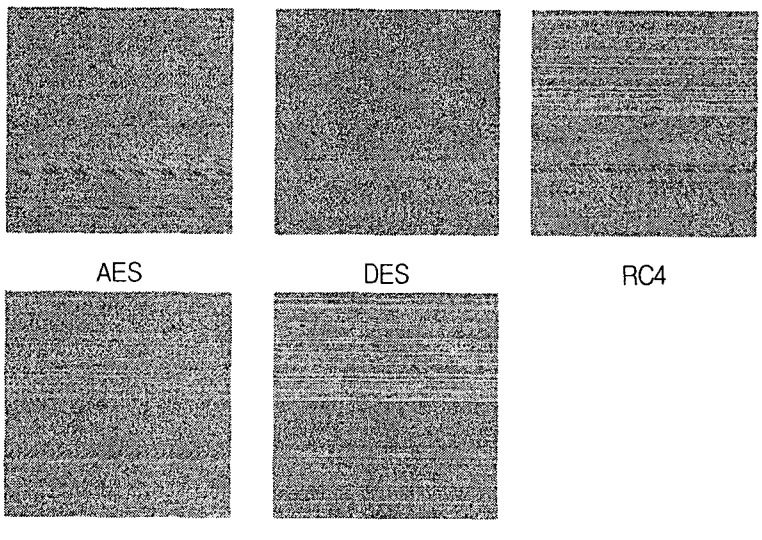
ciphertext를 이미지로 표현할 경우 특정한 형태의 패턴이 나타나는 문제는 다른 암호화 알고리즘을 이용하여 다시 암호화 한다 하더라도 해결되지 않는다.



RC4 / TDES

(그림 3) RC4와 TDES로 암호화된 ciphertext를 이미지로 표현한 경우

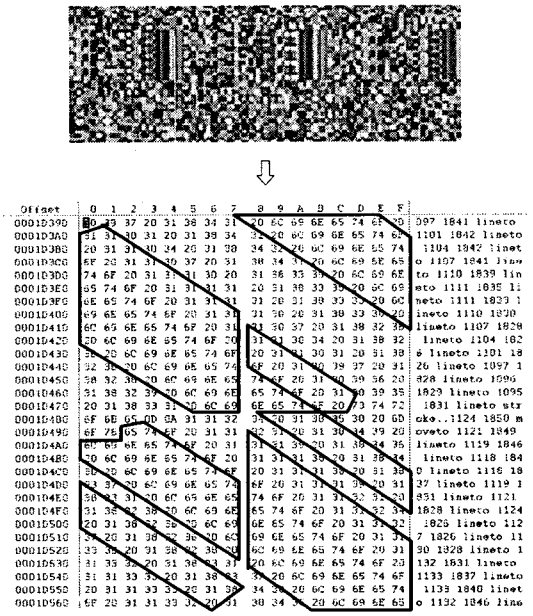
다음은 특정한 패턴이 나타나는 일부분의 이미지와 이에 대응하는 16진수 코드값이다.



AES DES RC4

SEED TDES

(그림 2) 암호화된 ciphertext를 이미지로 표현한 경우



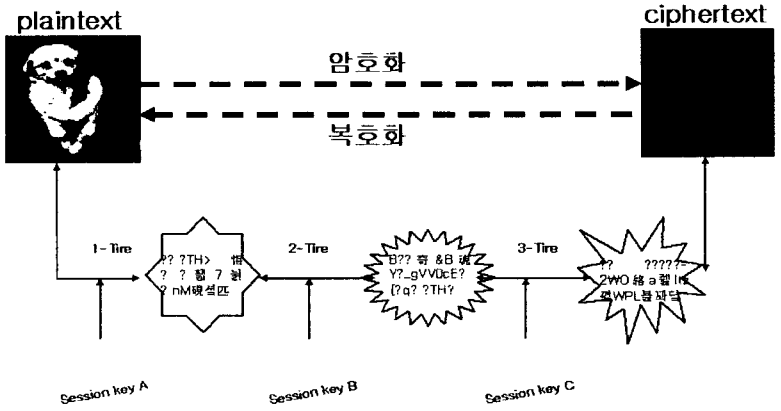
(그림 4) 패턴과 16진수 코드 값

위와 같이 chiphertext에 특정한 패턴이 나타난다는 문제점을 해결하기 위해 제안하는 모델에서는 새로운 블록 암호화 알고리즘인 MBE (Modyfied Block Encryption) 알고리즘을 제안하고 적용한다. MBE(Modyfied Block Encryption) 알고리즘은 암호화된 ciphertext의 구조와 성격을 은닉할 수 있도록 설계된 알고리즘이다. 또한 블록 암호화 알고리즘 적용 시 발생하는 padding 처리를 위해 SELI(Select Insert) padding 기법을 제안하고 적용한다.

3.2.1 1-Tire Stream 암호화

파일 암호화를 위해 1차적으로 암호화를 수행하는 단계이다. 적용할 수 있는 알고리즘은 스트림 암호화 알고리즘과 블록 암호화 알고리즘 모두 가능하지만 제안 모델에서는 최초의 plaintext 암호화 시 발생할 수 있는 오류의 단위를 최소화하기 위해 스트림 암호화 알고리즘을 적용한다. 모델에 적용된 알고리즘은 RC4이다.

RC4는 256바이트 이하의 길이를 가지는 가변키를 사용하는 알고리즘이다. plaintext를 암호화하기 위해 사용되는 의사-난수를 생성하기 위해 키를 기준으로 256바이트의 테이블을 초기화한다. 의사-난수를 생성하기 위해선 바이트당 8에서 16회까지의 연산이 필요하지만 매우 빠르게 처리가 가능한 알고리즘이다.[13][14]



(그림 5) GSF 파일을 이용한 파일 암호화 모델

SIZE : 256 ((1<<ALPHA) = (1 times 2 to the 8th))
ind(x) : the low order 8 bits of x, or x mod 256.
ALPHA : 8
SIZE : 1<<ALPHA
ind(x) : (x & (SIZE-1))
m : array of SIZE ALPHA-bit terms
r : the sequence, same size as m
aa : a single value

1. *For i from 0 to i<size*
   s←m[i]
   a←ind(a+x)
   y←m[a]
   m[i]←y
   m[a]←x
   r[i] ← m[ind(x+y)]

2. *aa←a*

(그림 6) RC4 알고리즘

3.2.2 SELI(Select Insert) padding 기법

ECB 모드와 CBC 모드로 블록 암호화 수행 시 전체 plaintext 크기가 블록의 배수가 되지 않을 경우 마지막 블록에 정형화된 데이터를 입력하는 padding을 수행하게 된다. 정형화된 데이터를 이용한 padding은 일정한 패턴을 가지는 값으로 암호화가 수행되므로 쉽게 padding된 영역을 찾을 수 있다는 단점을 가지게 된다. 제안하는 SELI padding에서는 실제 데이터가 입력된 블록의 특정 위치값을 padding 영역에 삽입한다. 이는 마지막 블록에 위치한 데이터가 일률적으로 padding된 데이터가 아니라 실제 데이터 값이 삽입되므로 padding 데이터뿐만 아니라 padding 수행여부도 알지 못하도록 하는 효과를 얻을 수 있다. 다음은 제안한 SELI padding 기법의 기본 알고리즘이다.

n : count of block
Pad_{byte} : padding byte count of last block
$NPad_{byte}$: real data byte count of last block
k : insert position
Value(i) : real value of *i* byte

1. $value((n-1)*8+ NPad_{byte})$
   $\leftarrow value((n-1)*8+ NPad_{byte}-1)$,
2. $value((n-1)*8+ NPad_{byte}-1) \leftarrow Pad_{byte}$
3. For *i* from *0* to $Pad_{byte}-1$
   $value((n-1)*8+NPad_{byte}+1+i)$
   $\leftarrow value(i*8+k)$

(그림 7) SELI padding 알고리즘

3.2.3 2-Tire Block 암호화

1-Tire에서 암호화된 ciphertext를 다시 암호화 하는 단계로 암호화 수준을 한 단계 더 높이기 위해 대체(substitution)와 치환(permutation)이라는 기본적인 암호화를 16번 반복하여 적용하는 DES 알고리즘을 이용하는 TDES 블록 암호화 알고리즘을 적용한다. TDES 블록 암호화 알고리즘은 DES를 2개의 서로 다른 키로 3번 암호화하는 알고리즘이다. TDES의 암/복호화는 다음과 같다.

Triple DES 암호화 : $c=E_{k1}(D_{k2}(E_{k1}(m)))$ (3-1)

Triple DES 복호화 : $m=D_{k1}(E_{k2}(D_{k1}(m)))$ (3-2)

DES 알고리즘은 입력을 좌 우 블록으로 나눈 후 한 블록을 라운드 함수에 적용한 후 결과값을 다른 블록에 적용시키는 과정을 좌우 블록에 대해 반복적으로 수행하는 Feistal 구조를 채택한 64비트 알고리즘이다. 하나의 블록에서 56비트가 키이고 나머지 8비트는 검사용으로 이용된

Input : plaintext $m_1...m_{64}$; 64bit key $K = k_1...k_{64}$

Output : 64bit ciphertext block $C = c_1,...c_{64}$

다. 다음은 기본적인 DES 알고리즘이다.

[6][7][10][15]

1. (key schedule) Computer sixteen 48-bit round keys K_i from K
2. $(L_0,R_0) \leftarrow IP(m_1...m_{64})$ $(L_0 = m_{58}m_{50}, ...m_8, , R_0 = m_{57}m_{49}, ...m_7)$
3. For i from 1 to 16
   $f(R_{i-1}, k_i) = P(S(E(R_{i-1}) \oplus K_i))$

   // DES inner function f
   (a) Expand $R_{i-1} = r_1r_2...r_{32}$ fro
   $T \leftarrow E(R_{i-1})$ (Thus T =
   (b) $T' \leftarrow T \oplus K_i$
   (c) $T'' \leftarrow (S_1(B_1),S_2(B_2),...S_8$
   (d) $T''' \leftarrow P(T'')$

4. $b_1b_2...b_{64} \leftarrow (R_{16}, L_{16})$
5. $C \leftarrow IP^{-1}(b_1b_2..b_{64})$

(그림 8) DES 알고리즘

3.2.4 3-Tire MBE

이미지 표현 시 발생할 수 있는 특정한 패턴을 제거하기 위해 2-Tire에서 암호화된 ciphertext의 구조와 형식을 은닉하는 단계이다. ciphertext 블록들의 난수를 생성한 후 이를 기반으로 한 키를 이용하여 ciphertext를 암호화하는 MBE (Modyfied Block Encryption) 알고리즘을 구현하였다.

제안된 알고리즘은 이미지 키로부터 생성해낸 255개의키를 순차적으로 순환하며 각 블록과 XOR 연산을 수행한다. XOR 연산 수행 시 앞 블록의 결과 데이터가 다음 블록의 결과에 영향을 미치도록 구현되어 있기 때문에 ciphertext의 구조와 형식을 은닉할 뿐만 아니라 한번의 암호화가 더 수행된다.

ciphertext의 3-Tire 암호화에서 사용되는 MBE 알고리즘은 다음과 같다.

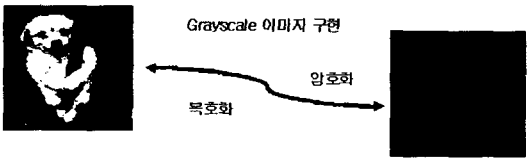
Input : $d_{(0)},...d_{(n)}$ created block data from plaintext
$k_{(0,...255)}$ created key from the table of secret message in numbers
Ouput : $o_{(0)}...o_{(n)}$ result calculated per block

1. $o_{(0)} \leftarrow d_{(0)} \oplus k_{(0)}$
2. For i from 1 to $n-1$
   $o_{(i)} \leftarrow d_{(i)} \oplus k_{(i \% 255)} \oplus o_{(i-1)}$

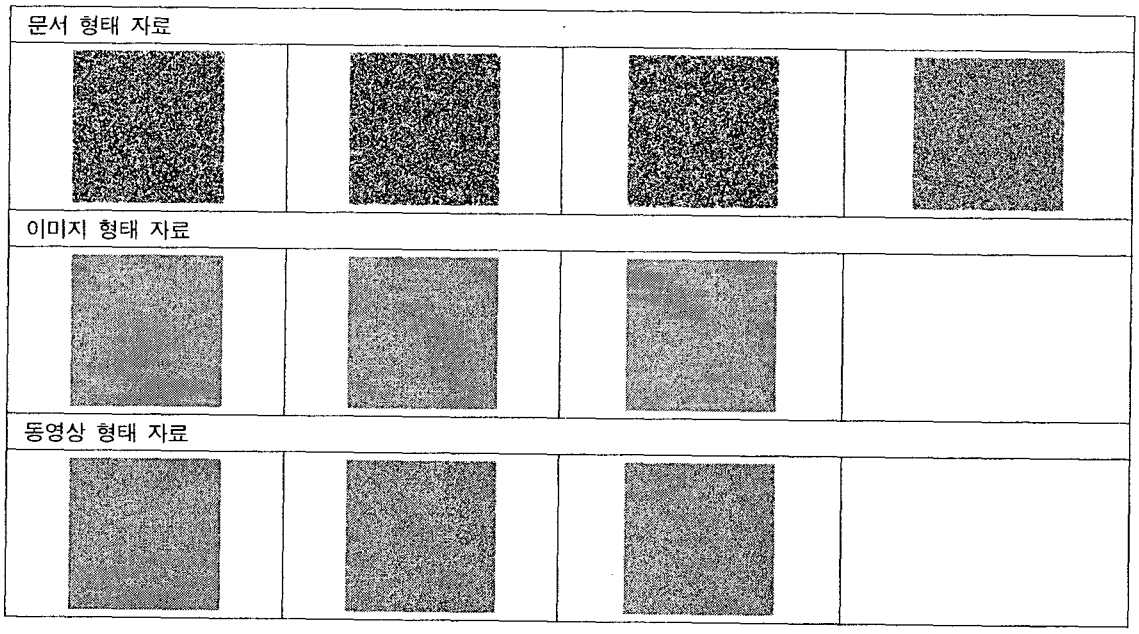
(그림 9) MBE 알고리즘

3.2.5 GSF 이미지 파일 생성 단계

제안한 알고리즘의 3-Tire 암호화 과정을 거친 ciphertext를 이미지형태로 구현하는 단계이다. ciphertext의 비트값을 대응되는 grayscale 값으로 변환하여 이미지 파일을 생성한다. 많은 다양한 데이터를 사용하여 GSF 파일로 생성해 보면 GSF들이 서로 유사하게 보여 원본의 차별성을 인식할 수 없다.



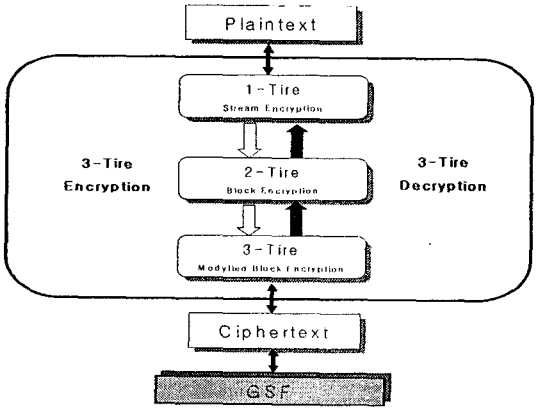
(그림 10) GSF 파일 생성단계



(그림 11) 다양한 데이터 파일에 적용한 GSF 결과

IV. 시스템 구현

제 3장에서 살펴본 GSF 출력을 이용한 3-Tire 파일 암호화 알고리즘의 구현을 위한 환경을 기술하고, 실제 구현된 모델을 이용하여 다양한 유형의 데이터가 암호화된 이미지를 생성한다.



(그림 12) 시스템 구성도

4.1 구현

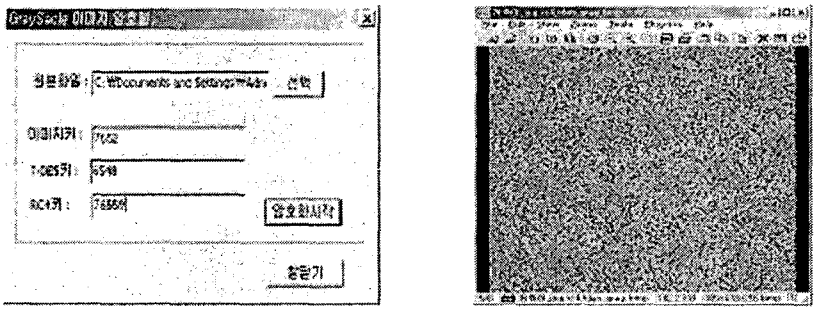
운영체제는 Windows 2000을 사용하였으며 프로그래밍 언어는 Visual Studio.Net을 이용하여 구현하였다. 또한 사용된 plaintext는 text 파일, application 파일, image 파일, moving picture 파일로 각각 서로 다른 크기를 가지는 세 개 파일을 이용하였다.

구현된 모델에서는 1-Tire Stream 암호화, 2-Tire Block 암호화, 3-Tire MBE, 이미지 생성의 과정을 따로 처리하지 않고 한번에 필요한 모든 킷값을 입력받은 후 순차적으로 암호화 및 이미지 파일이 생성되도록 구현하였다. 이미지

구현 시 grayscale 이미지뿐만 아니라 color 이미지도 생성되도록 구현하였다.

데이터 암호화 및 grayscale 이미지 생성

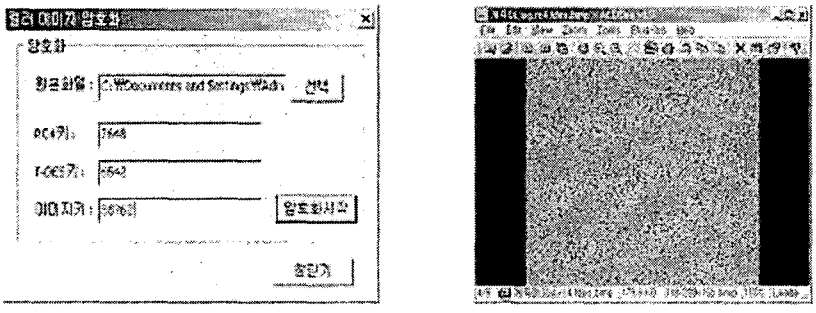
원본 파일명과 실제 암호화에 사용되는 세션 키를 직접 입력받도록 구현하였다. RC4 알고리즘은 4자리 이상의 키 값을 입력받도록 하였다. TDES는 키의 길이가 4자리 이상이 되는 4의 배수값을 입력하도록 하였으며 image에 사용되는 키는 4자리 이상의 값으로 지정하도록 제한하였다.



(그림 13) 데이터 암호화 및 grayscale 이미지 생성

데이터 암호화 및 color 이미지 생성

암호화된 ciphertext를 2바이트씩 16비트 컬러 값으로 사용하여 컬러 이미지로 구현하였다.



(그림 14) 데이터 암호화 및 color 이미지 생성

V. 결론

본 연구에서는 기존의 암호화 프로그램들이 ciphertext를 텍스트파일 형태로 표현하기 때문에 누구나 암호화된 파일이라는 것을 알 수 있다는 문제점을 해결하고 Steganography와는 달리 cover 이미지를 사용하지 않고 이미지 파일로 표현할 수 있는, 새로운 파일 암호화 알고리즘을 제안하고 구현하였다. 제안한 GSF 출력을 이용한 파일 암호화 알고리즘에서는 기존에 제안된 알고리즘을 이용하여 2-Tire 암호화를 수행한 뒤 ciphetext를 이미지로 표현시 특정한 패턴이 발생하는 문제점을 해결하기 위해 새로운

블록 알고리즘인 MBE 알고리즘을 적용하여 새로운 형태의 암호화를 한 번 더 수행하였다. 또한 블록암호화 수행 시 제안한 SELI padding을 적용함으로써 padding 데이터의 생성뿐만 아니라 padding 수행 여부도 판별하기 어렵도록 지원하였다. 제안한 GSF 출력을 이용한 3-Tire 파일 암호화 알고리즘을 적용하여 생성된 ciphertext는 원본 데이터 파일의 종류에 관계없이 모두 거의 비슷한 패턴을 가지는 이미지파일이기 때문에 파일의 암호화뿐만 아니라 ciphertext의 은닉 효과까지 기대할 수 있다.

향후에는 외부의 공격에 충분히 안정성을 가지며 암호화된 ciphertext를 좀 더 효과적으로 은닉할 수 있는 최적화 파일 암호화 알고리즘의 구현이 필요하다.

참고문헌

[1] 박창섭, "암호이론과 보안", pp.13-83, 대영사, 1999

[2] 이동훈, 임채훈, "국제/업계 표준 암호 알고리즘 및 프로토콜의 이해", pp.2-10, (주)퓨처시스템 암호체계센터, 2000

[3] 김현곤, 원동호 외 12, "지적재산권 보호를 위한 정보은닉 기술 및 표준화 연구", pp. 19-41, 한국 전산원 2000

[4] 이종근, 최돈승, 이경석, "정보보호를 위한 암호운영방식 고찰", pp 13-18, 창원대학교 정보통신연구소 논문집 제 2집, 1998

[5] Ross J Anderson, Fabien A.P. Petitcolas "On The Limits of Steganography" pp. 474-481, IEEE Journal on Selected Areas in Communications. 1998

[6] Bruce Schneier, "Applied Cryptography", 1996, pp.265-301,

[7] Raymond G. Kammer, "DATA ENCRYPTION STANDARD", Federal Information Processing Standards Publication, 1999

[8] http://www.networksorcery.com/enp/data/encryption.htm

[9] http://www.netaction.org/encrypt/appendixb.html

[10] http://www.networksorcery.com/enp/rfc/rfc3217.txt

[11] http://www.securitytechnet.com

[12] http://www.itl.nist.gov/fipspubs/fip81.htm

[13] http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html

[14] http://burtleburtle.net/bob/rand/isaac.html

[15] http://www.cerberussystems.com/INFOSEC/stds/fip46-3.htm

3-Tire File Encryption algorithm using GSF

Young-Shil, Kim* · Young-Mi, Kim** · Ryun-Ok, Kim*** · Doo-Kwon, Baik***

Abstract

This paper proposes improved file encryption algorithm which represents image of grayscale type not using proper cover image for ciphertext. This method consists of 3-Tire encryption steps. 1-Tire and 2-Tire encrypt the information using existed stream algorithm and block algorithm with modyfied padding method. We propose the MBE method as 3-Tire, which hides structure and format of encrypted file. The proposed method outputs grayscale file as the result of encryption and since many GSF outputs resulted from different kinds plaintexts, have similar patterns. we obtain both file encryption and hiding the file information. Also, to solve the problem of padding in block algorithm, we propose the new padding algorithm called SELI(Select Insert) and apply 2-Tire block algorithm and MBE algorithm used 3-Tire.

* Dept of Computer Science & Information, Daelim College
** Dept of Development, CEST CO.LTD
*** Dept of Computer Science & Information, Daelim College
**** College of Information & Communication, Korea University