# 스테가노 그래피
# 헤더를 포함한 암호화된 데이터 은닉

https://youtu.be/8Te7Qf3tf-g

# 스테가노 그래피

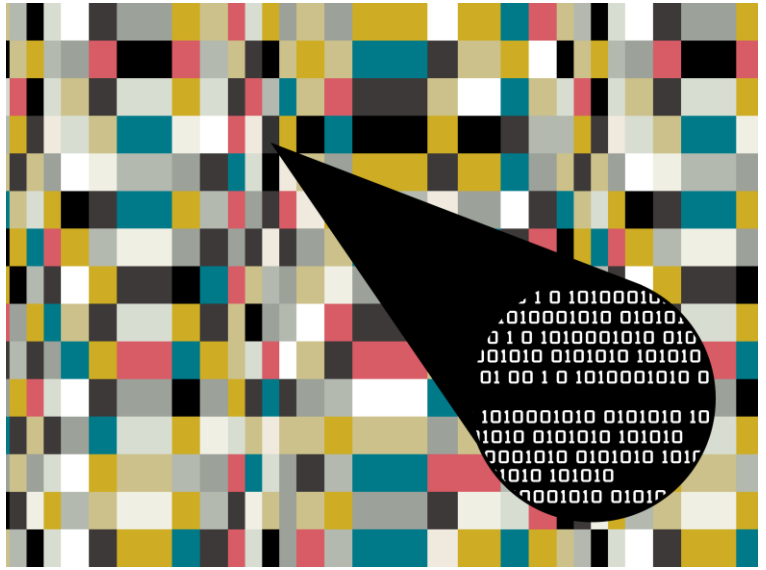
- 데이터 은폐 기술의 하나이며 데이터를 다른 데이터에 삽입하는 기술
- 어느 정도 보안을 유지해 주는 기능을 갖고 있기 때문에, 오랜 기간 동안 사용
- 스테가노 그래피는 데이터를 은닉할 뿐 기밀성을 보장하지 않는다

  실전에서는, 스테가노그라피 와 암호화 기법을 함께 사용

  메시지를 먼저 암호화 하여 스테가노그라피를 사용해 숨겨서 전달

- 디지털(Digital) 스테가노그라피

  텍스트 파일, 그림(image) 파일, 오디오 파일, 동영상



# 스테가노 그래피의 목표

- 얼마나 임베딩 할 수 있는가?
- 표지 이미지와 품질, 이미지의 크기, 지각적으로 동일한가?
- 스테가노 분석에 저항 할 수 있는가?

✓ 이상적인 스테가노그래피 방법은 위의 목표를 고용량으로 동시에 달성

# 제안 기법

- 스테가노 그래피에 암호화된 데이터를 임베딩 할 때 효율적인 방법

➢ 형태 보존 암호를 사용한 해더 + 데이터 구조의 암호화를 제안
  - ✓ 은닉 대상 데이터의 크기 감소
  - ✓ 스테가노 분석에 저항을 위해 데이터의 분할과 은닉 위치를 자유롭게 선택가능

# 관련 논문

## Data Hiding for Ensuring the Quality of the Host Image and the Security of the Message

**HUAIBO SUN, HONG LUO, AND YAN SUN**
Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China
Corresponding author: Huaibo Sun (sunhuaibo@ bupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772085, Grant 61877005, and Grant 61672109.

**ABSTRACT** Concealing a message in the APPn markers of a JPEG image not only protects the security of the message but also induces no change to the quality of the image. However, from the existing literature, whether it is a plaintext message or a ciphertext message hidden in APPn, it is easy for an attacker to identify the confidentiality of the hidden message, which is not conducive to the security of the message. Inspired by the natural language processing (NLP) and format-preserving encryption (FPE), this paper proposes a data hiding method, which is focused on the quality assurance of the host image and the concealment of the plain text having complete semantics based on the NLP and FPE. This method first uses the NLP and FPE to identify and encrypt the sensitive words in the plain text and then hides the ciphertext text with the plaintext style in the APPn of a JPEG image after replacing the plaintext words with the ciphertext words that have the plaintext style. The experimental results confirm that the structure, size, and quality of the host image do not show any changes before or after the data hiding and the recovered host image is also identical to its original appearance. In addition, the strategy that the semantic similarity is regulated autonomously by the user also makes it possible to obtain the ideal ciphertext words with very low similarity. Moreover, more than 80% of the ciphertext texts have reasonable semantics. Compared with the existing literature, our algorithm has a better performance.

# 관련 논문

JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 22, No. 5: 805-811, May 2018

## 형태보존암호화를 이용한 랜섬웨어 방지 및 스테가노그래피 보안 강화기술

임지환[1] · 나관우[1] · 우재민[1] · 서화정[1*]

## Ransomware Prevention and Steganography Security Enhancement Technology Using Format Preserving Encryption

Ji-hwan Lim[1] · Gwan-Woo Na[1] · Jae-Min Woo[1] · Hwa-joeng Seo[1*]

[1*]Department of IT Engineering, Hansung University, Seoul 02876, Korea

요 약

형태 보존 암호는 암호화하고자 하는 목적 정보에 대한 변형 없이 형태를 유지한 상태로 암호화하는 기법으로써 최근에 국가보안기술연구소에 의해 제안되었다. 본 논문에서는 형태 보존 암호를 활용하여 기존의 사이버 보안 관련 문제를 해결하는 방안을 제안하고자 한다. 먼저 랜섬웨어 공격을 효과적으로 방어하기 위해 시그니처 및 확장자를 형태보존암호로 암호화하는 방안을 제시한다. 해당 기법은 최소한의 정보를 암호화함으로써 랜섬웨어에 대한 노출을 최소화할 수 있다. 두 번째로 스테가노그래피와 같이 비밀 정보를 숨기는 기술상에서도 해당 정보의 양을 최소화함으로써 공격에 대비할 수 있는 방안을 제시한다. 마지막으로 형태보존암호와 경량암호에서 암호화에 따른 동작 속도를 비교하고, 형태보존암호를 최적화하였을 때, 그에 따른 성능 향상까지 비교하고자 한다.

# 관련 논문

2.2. 랜섬웨어 예방을 위한 확장자/시그니처 암호화

랜섬웨어의 동작 원리 중 하나는 특정한 확장자를 탐색하여 이를 암호화하는 방식이다. 그 이유는 모든 파일을 암호화하게 될 경우 시스템 동작이 불가능하기 때문이다. 따라서 이를 예방하기 위해 확장자를 암호화하여 파일의 속성을 숨긴다면 악성코드의 파일 탐지로부터 1차적으로는 벗어날 수 있다. 하지만 해당 기법 역시 파일이 가지는 고유 시그니처 정보가 변경되지 않아 여전히 해킹 위험이 남아있다. 표. 1을 보면 각각의 파일은 파일마다 특정한 시그니처가 있는데, 핵심은 확장자와 시그니처를 동시에 형태보존암호를 사용하여 암호화해야 안전하다는 것이다. 확장자/시그니처 암호화에 형태보존암호를 제안한 이유는 [확장자명||시그니처]를 형태보존암호화 시키면 평문과 동일한 길이로 암호화가 가능해서 다른 암호화에 비해 파일 암·복호화, 관리 등이 수월하고 공격자가 파일의 암호화를 확인하기 어렵다는 특징이 있다. 암호화에 따른 속도 비교는 4장에서 하도록 하겠다. 본 논문의 2.2.1.과 2.2.2.에서는 제안하고자 하는 방법에 대한 2가지 시스템 동작 메커니즘을 설명하겠다.

**Table. 1** Signature information for specific file [9]

| Extension | Signature | Offset | Explanation |
|---|---|---|---|
| AVI | 52 49 46 46 | 0 | Video File |
| BMP | 42 4D | 0 | Image File |
| GIF | 47 49 46 38 | 0 | Image File |
| PDF | 25 50 44 46 | 0 | PDF File |

# 관련 논문

LEA와 FEA로 8MB의 GIF 파일을 암호화했을 때 데이터베이스 포맷을 유지한다는 전제 하에 FEA는 [확장자||시그니처]의 7byte만 암호화 하면 되지만, LEA는 [확장자||시그니처]가 15byte 이하라면 데이터베이스 포맷을 유지하기 위해 8MB를 전부 암호화해야 한다. 그에 따른 성능 테스트는 표. 4와 같다. 정리하면 LEA는 8MB 파일을 암·복호화 하는데 13.143sec 시간이 소요되고, FEA는 7B를 암·복호화 하는데 $2.625*10^{-5}$sec 시간이 소요됨으로, FEA로 암·복호화 했을 때 500,000 배 이상의 속도가 나온다. 랜섬웨어 대상인 파일 대부분의 [확장자||시그니처]가 15byte 이하인 점을 고려하면 이는 매우 안전하고 효율적이다.

**Table. 4** Comparison of LEA (8MB encrypt and decrypt) and FEA (7B encrypt and decrypt)

| Method | LEA | FEA |
|---|---|---|
| Timing | 13.143s | $2.625\times10^{-5}$s |

# 관련 논문

2.3. 스테가노그래피 보안강화기술

본 절에서는 형태보존암호를 활용하여 기존 스테가노그래피 정보 삽입의 문제점을 보완할 수 있는 방법을 제안한다. 스테가노그래피 암호화 기법은 많은 정보를 숨기게 될 경우 원본에 대한 변형이 크게 일어나 해커가 이를 눈치 채고 탐지가 보다 쉽게 일어난다는 문제점이 있다. 기존 블록 암호의 가장 큰 문제점은 암호화 후에 스테가노그래피에 들어가는 데이터 크기이다. 형태보존암호는 암호화 전후에 데이터의 크기가 변하지 않지만, 다른 블록 암호는 16byte 단위로 암호화함으로 대부분 경우에 데이터의 크기가 늘어난다. 즉, 스테가노그래피에 들어가는 정보가 많아지게 되고 정보의 변형이 더 커지게 된다. 이는 정보 변형이 민감한 스테가노그래피 보안기법에서 비효율적이다. 또한, 형태보존암호의 이점으로 정보 변형 크기가 같은 수준에서 암호화할 데이터를 더 많이 삽입할 수 있다. 성능평가의 2.25MB 비트맵 방식의 이미지에서 각 픽셀의 LSB에 데이터를 삽입하는 예시를 통해 결과를 확인할 수 있다.

**Table. 5** Comparison of LEA and FEA 2.25MB bitmap image file encryption and decryption

| Method | LEA | FEA |
|---|---|---|
| Timing | 0.18s | 0.4s |

# TEXT를 형태 보존 암호로 암호화

TEXT

- 암호화

Base64 인코딩 -> 암호화

a~z -> 0~25 -> 암호화

: 한 문자에 비트 5 bit로 인코딩

복호화 시 인코딩 방식에 대하여 알아야함

➢ 추가정보를 헤더에 넣는다면 해당 정보는 드러남
탐지가 쉬움

✓ 따라서 항상 헤더 부분의 크기는 고정하여 보내며
헤더 부분을 암호화

✓ 데이터 부분은 형태보존 암호를 사용하여 암호화 한다면 짧게
끊어 저장 가능

제어 문자 | 공백 문자 | 구두점 | 숫자 | 알파벳

| 10진 | 16진 | 문자 | 10진 | 16진 | 문자 | 10진 | 16진 | 문자 | 10진 | 16진 | 문자 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0x00 | NUL | 32 | 0x20 | SP | 64 | 0x40 | @ | 96 | 0x60 | ` |
| 1 | 0x01 | SOH | 33 | 0x21 | ! | 65 | 0x41 | A | 97 | 0x61 | a |
| 2 | 0x02 | STX | 34 | 0x22 | " | 66 | 0x42 | B | 98 | 0x62 | b |
| 3 | 0x03 | ETX | 35 | 0x23 | # | 67 | 0x43 | C | 99 | 0x63 | c |
| 4 | 0x04 | EOT | 36 | 0x24 | $ | 68 | 0x44 | D | 100 | 0x64 | d |
| 5 | 0x05 | ENQ | 37 | 0x25 | % | 69 | 0x45 | E | 101 | 0x65 | e |
| 6 | 0x06 | ACK | 38 | 0x26 | & | 70 | 0x46 | F | 102 | 0x66 | f |
| 7 | 0x07 | BEL | 39 | 0x27 | ' | 71 | 0x47 | G | 103 | 0x67 | g |
| 8 | 0x08 | BS | 40 | 0x28 | ( | 72 | 0x48 | H | 104 | 0x68 | h |
| 9 | 0x09 | HT | 41 | 0x29 | ) | 73 | 0x49 | I | 105 | 0x69 | i |
| 10 | 0x0A | LF | 42 | 0x2A | * | 74 | 0x4A | J | 106 | 0x6A | j |
| 11 | 0x0B | VT | 43 | 0x2B | + | 75 | 0x4B | K | 107 | 0x6B | k |
| 12 | 0x0C | FF | 44 | 0x2C | , | 76 | 0x4C | L | 108 | 0x6C | l |
| 13 | 0x0D | CR | 45 | 0x2D | - | 77 | 0x4D | M | 109 | 0x6D | m |
| 14 | 0x0E | SO | 46 | 0x2E | . | 78 | 0x4E | N | 110 | 0x6E | n |
| 15 | 0x0F | SI | 47 | 0x2F | / | 79 | 0x4F | O | 111 | 0x6F | o |
| 16 | 0x10 | DLE | 48 | 0x30 | 0 | 80 | 0x50 | P | 112 | 0x70 | p |
| 17 | 0x11 | DC1 | 49 | 0x31 | 1 | 81 | 0x51 | Q | 113 | 0x71 | q |
| 18 | 0x12 | DC2 | 50 | 0x32 | 2 | 82 | 0x52 | R | 114 | 0x72 | r |
| 19 | 0x13 | DC3 | 51 | 0x33 | 3 | 83 | 0x53 | S | 115 | 0x73 | s |
| 20 | 0x14 | DC4 | 52 | 0x34 | 4 | 84 | 0x54 | T | 116 | 0x74 | t |
| 21 | 0x15 | NAK | 53 | 0x35 | 5 | 85 | 0x55 | U | 117 | 0x75 | u |
| 22 | 0x16 | SYN | 54 | 0x36 | 6 | 86 | 0x56 | V | 118 | 0x76 | v |
| 23 | 0x17 | ETB | 55 | 0x37 | 7 | 87 | 0x57 | W | 119 | 0x77 | w |
| 24 | 0x18 | CAN | 56 | 0x38 | 8 | 88 | 0x58 | X | 120 | 0x78 | x |
| 25 | 0x19 | EM | 57 | 0x39 | 9 | 89 | 0x59 | Y | 121 | 0x79 | y |
| 26 | 0x1A | SUB | 58 | 0x3A | : | 90 | 0x5A | Z | 122 | 0x7A | z |
| 27 | 0x1B | ESC | 59 | 0x3B | ; | 91 | 0x5B | [ | 123 | 0x7B | { |
| 28 | 0x1C | FS | 60 | 0x3C | < | 92 | 0x5C | \ | 124 | 0x7C | \| |
| 29 | 0x1D | GS | 61 | 0x3D | = | 93 | 0x5D | ] | 125 | 0x7D | } |
| 30 | 0x1E | RS | 62 | 0x3E | > | 94 | 0x5E | ^ | 126 | 0x7E | ~ |
| 31 | 0x1F | US | 63 | 0x3F | ? | 95 | 0x5F | _ | 127 | 0x7F | DEL |

Base64 색인표

| 값 | 문자 | 값 | 문자 | 값 | 문자 | 값 | 문자 |
|---|---|---|---|---|---|---|---|
| 0 | A | 16 | Q | 32 | g | 48 | w |
| 1 | B | 17 | R | 33 | h | 49 | x |
| 2 | C | 18 | S | 34 | i | 50 | y |
| 3 | D | 19 | T | 35 | j | 51 | z |
| 4 | E | 20 | U | 36 | k | 52 | 0 |
| 5 | F | 21 | V | 37 | l | 53 | 1 |
| 6 | G | 22 | W | 38 | m | 54 | 2 |
| 7 | H | 23 | X | 39 | n | 55 | 3 |
| 8 | I | 24 | Y | 40 | o | 56 | 4 |
| 9 | J | 25 | Z | 41 | p | 57 | 5 |
| 10 | K | 26 | a | 42 | q | 58 | 6 |
| 11 | L | 27 | b | 43 | r | 59 | 7 |
| 12 | M | 28 | c | 44 | s | 60 | 8 |
| 13 | N | 29 | d | 45 | t | 61 | 9 |
| 14 | O | 30 | e | 46 | u | 62 | + |
| 15 | P | 31 | f | 47 | v | 63 | / |

# 이미지를 형태 보존 암호로 암호화

Image

한 픽셀

- RGB(0~255, 0~255, 0~255) 24 bit
- GRAY(0~255) 8 bit
- **텍스트와 동일한 기법으로 크기를 줄여 암호화 불가능**

➢ 이미지의 색 범위가 0~255보다 작다면?

- 텍스트와 동일한 기법으로 크기를 줄여 암호화 가능

128~255 -> 0~128  7bit로 가능

-> 복호화 시 범위를 알아야함

# 이미지를 형태 보존 암호로 암호화



복호화 시

- **색의 범위 정보**
- **나누어진 위치 정보**
  - 이 두가지 정보도 함께 전달 필요
  - 해당정보는 이미지의 특징을 담고 있으므로 암호화되어야 함

# 이미지를 형태 보존 암호로 암호화

- (헤더) + (데이터) 구조로 암호화

- 헤더 :
  색의 범위 (범위) 2bit
  0~255 : 11,  0~127 : 10,  0~64 : 01,  0~32 : 00
  데이터 길이  (블록 개수) 6 bit
  - 헤더 작은 데이터의 길이 : 형태보존 암호 사용 적합

- 데이터 :
  특정 범위 안에 포함되는 색상 픽셀

# 한계점

- 이미지의 경우 효율적인 압축기법들이 많이 있음

- 추후에는 압축기법과 스테가노 그래피를 같이 사용할 수 있는 기법에 대하여 연구 예정

# Q & A