노노그램을 사용한 이미지 암호화

Image-to-Text Encryption Using Nonogram

조훈민 Hoonmin Cho
성균관대학교 Sungkyunkwan University
dkwkgnsals@gmail.com

정문수 Moonsoo Jeong
성균관대학교 Sungkyunkwan University
moonsoo101@g.skku.edu

김종민 Jongmin Kim
성균관대학교 Sungkyunkwan University
kimjmin207@skku.edu

이성길 Sungkil Lee
성균관대학교 Sungkyunkwan University
sungkil@skku.edu

요약

본 논문은 이미지 암호화 연구에 관한다. RGB 로 구성된 이미지 파일의 경우 R, G, B 값을 2 진수로 변환했을 때 나타나는 8 개의 자릿수 값으로 총 24 개의 비트맵으로 표현가능하고, 이 비트맵은 노노그램(Nonogram)을 사용해서 텍스트 파일로 변환가능하다. 이에 본 논문에서는 노노그램을 사용해서 image-to-image 암호화 방식이 아닌 image-to-text 암호화 기법을 제안한다.

주제어

이미지 암호화, 노노그램

1. 서 론

타인의 정보를 해킹하는 기술이 나날이 발전하고 있는 요즘, 데이터 전송에 있어 데이터를 암호화 하여 해커에게 보여져서는 안되는 정보를 보호하는 것은 반드시 적용해야 할 필수 기술로 자리잡고있다. 그 중에서도 이미지 데이터의 암호화 기법은 여러가지가 제안되었지만, 그 대부분이 하나의 이미지 데이터를 Key 값을 사용해서 본래의 이미지를 알 수 없는 다른 이미지 데이터로 변환시키는 방법을 사용하고 있다.

본 논문은 기존의 방법과 다르게 이미지 데이터를 텍스트 데이터로 변환시키기 위하여, 노노그램[1]을 사용하는 알고리즘을 제안한다. 입력 이미지의 각 픽셀은 RGB 값에 따라서 8 자리의 2 진수값 3 개로 변환 가능하고, 각 자릿수 값을 사용하면 입력 이미지와 같은 해상도를 가지는 24 개의 비트맵을 생성할 수 있다. 각각의 비트맵은 0 을 빈 칸으로, 1 을 채워진 칸으로 해석하여 노노그램을 적용시키면 24 개의 비트맵을 만들 수 있는 24 쌍의 텍스트 파일을 생성할 수 있다. 본 논문에서는 이중 암호화를 적용하기 위해서 24 쌍의 텍스트 파일을 하나로 연결하여 표현하는 방법과 24 개의 비트맵을 배열하는 경우의 수를 사용하는 방법을 사용하였다.

2. 관련연구

2.1 Image-to-image 암호화기법

이미지를 다른 이미지로 암호화 하는 기법에는 Chaotic system 을 기본으로 하는 연구가 진행되었다. 한 연구에서는 Chaotic system 을 사용해서 암호화 전과 암호화 후의 이미지의 히스토그램이 거의 동일하게 나타나게 하는 기법을 제안하였고[2], 또 다른 연구에서는 Chaotic system 에 DNA computing 기술까지 사용하여 이미지를 암호화 하는 기법을 제안했다[3]. 하지만 두 연구 모두 이미지를 이미지로 암호화 했다는 점에서 본 논문의 기법과 차이가 있다.

2.2 노노그램 문제 해결방법

본 논문에서 제안하는 기법은 암호화된 데이터를 본래의 이미지로 복호화 할 경우 노노그램을 정확히 풀어야 한다는 것이 전제되어 있다. 노노그램을 해결하는 알고리즘을 찾기 위한 연구는 이미 여러가지 진행되어 왔고, 컴퓨터상에서 노노그램을 해결하기 위해서 발견된 알고리즘을 프로그램으로 제작하려는 연구도 다수 진행되었다. 본 논문에는 암호화된 데이터를 복호화하기 위해서 이미 연구된 노노그램 해결 알고리즘을 사용한다.

3. 알고리즘

본 논문에서는 그림 1 과 같이 이미지 데이터를 텍스트 데이터로 변환시킴으로써 암호화 시키고 이를 다시 원본 이미지로 복호화하는 과정을 소개한다.

3.1 이미지 데이터를 텍스트 데이터로 변환

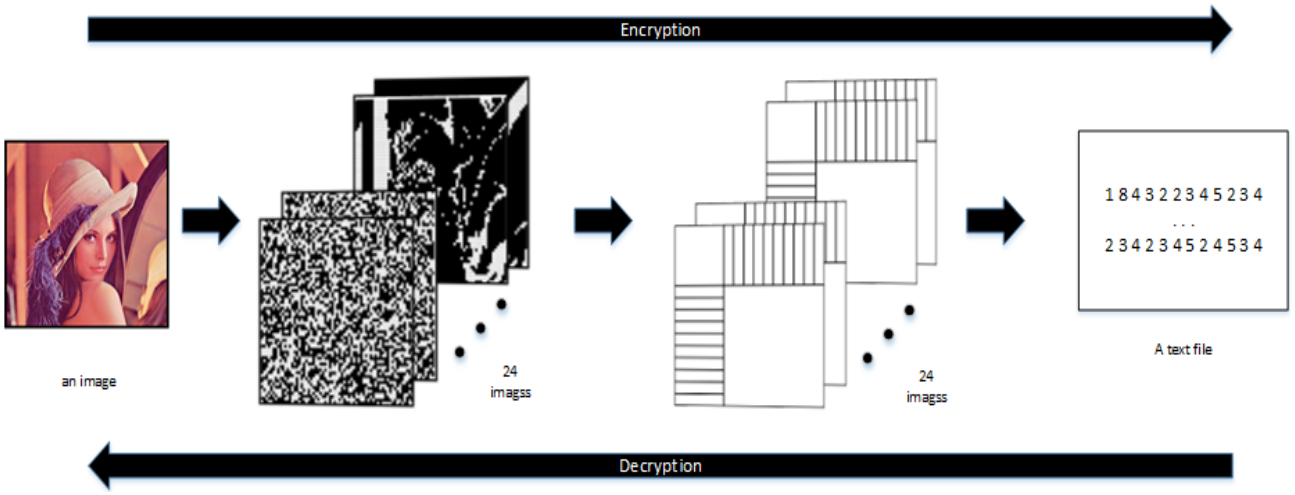

그림 1 이미지 암호화 및 복호화 전 과정의 개요도

본 논문의 기법은 최종적으로 노노그램을 사용해 이미지 데이터를 복원하기 위한 텍스트 데이터를 얻기 위해서 원본 이미지의 RGB 값을 기반으로 비트맵을 생성한다. RGB 값은 각각 0 에서 255 의 크기를 가지고 있으며 이는 2 진수를 사용해서 각 자릿수 마다 0 혹은 1 의 값을 가지는 8 자리 숫자로 나타낼 수 있다. 2 진수로 나타낸 RGB 각각의 자릿수에서 0 은 빈칸으로, 1 은 채워진 칸으로 생각하여 비트맵 파일을 생성하면 그림 2 와 같은 비트맵 데이터를 각 채널마다 8 개씩, 총 24 개를 만들어낼 수 있다.

생성된 각각의 비트맵은 노노그램에서 비트맵 형식의 이미지를 여러 개의 숫자로 구성된 문제의 형태로 만드는 것처럼 본래의 비트맵 복원을 가능하게 하는 텍스트 데이터로 변환된다.

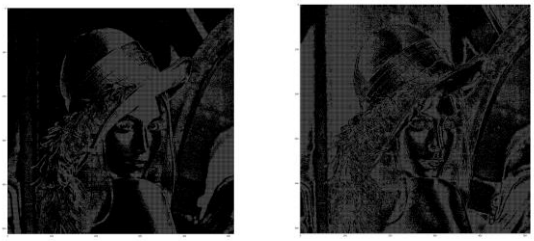

그림 2. 텍스트 데이터로 변환 전 비트맵 데이터

3.2 효율적인 복호화 및 다중 암호화를 위한 데이터 변환

노노그램을 사용해서 텍스트 데이터를 비트맵 데이터로 복호화하는 과정에 필요한 시간은 비트맵의 해상도가 증가할수록 기하급수적으로 증가한다. 따라서 본 논문은 시간적으로 더 효율적인 복호화와 다중 암호화를 위한 데이터 변환 기법을 제안한다.

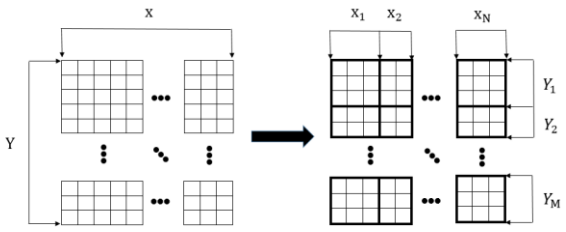

그림 3. 원본 비트맵의 데이터 분할

그림 3 처럼 원본 이미지 데이터로부터 생성된 비트맵의 가로 픽셀 수를 X 세로 픽셀 수를 Y 라고 가정하고 가로 영역을 N 개의 영역으로, 세로 영역을 M 개의 영역으로 나누면 하나의 비트맵은 총 $N*M$ 개의 영역으로 나눠지고 각 영역의 가로, 세로 픽셀 수는 각각 $X_1, X_2, \ldots, X_N$, $Y_1, Y_2, \ldots, Y_M$ 으로 나타낼 수 있다. 이렇게 생성된 $N*M$ 개의 비트맵은 노노그램을 사용해서 각각 다른 텍스트 데이터로 변환 가능하고 멀티스레딩을 사용한다면 이 데이터들을 복호화하는 작업은 원본 이미지 크기의 데이터를 복호화하는 작업에 비해서 큰 시간 절감 효과를 기대할 수 있다.

또한 $N*M$ 개의 텍스트 데이터를 하나의 데이터로 합치는 과정을 추가하여 데이터 전송 횟수를 한번으로 줄이고 $X_1, X_2, \ldots, X_N, Y_1, Y_2, \ldots, Y_M$ 값을 첫 번째 Key 로 사용해서 데이터의 암호화 효과를 가져온다. 오직 띄어쓰기와 숫자만을 사용하여 하나의 텍스트 데이터를 만드는 과정은 다음과 같다. 1~N 의 값을 가지는 a, 1~M 의 값을 가지는 b 를 사용해서 각 영역을 (a,b)로 표현하고 이 영역의 제일 왼쪽의 열부터 맨 오른쪽의 열까지, 그 다음에는 맨 위의 행부터 맨 아래의 행까지의 순서로 노노그램을 사용하여 비트맵 복원을 위한 숫자들을 구한 뒤

띄어쓰기를 통해 구분하면서 텍스트 데이터에 채워 넣는다. 이 때, 다음 행 또는 열 혹은 다음 영역의 숫자로 넘어가야 하는 경우에는 노노그램의 규칙을 사용하여 그 행 또는 열 안에 존재할 수 있는 가장 큰 숫자 더하기 1 의 값을 전달 신호로서 텍스트 데이터에 넣는다. 예를 들어 (a, b)의 영역을 텍스트 데이터에 넣는다고 가정하면, 넣어야 하는 숫자의 위치가 임의의 열에서 다른 열로 이동하거나 마지막 열인 경우에는 $X_a + 1$의 값을 텍스트 데이터에 넣고 임의의 행에서 다른 행으로 이동하거나 마지막 행인 경우에는 $Y_b + 1$의 값을 텍스트 데이터에 넣는다. 이 과정을 $(X_1 Y_1)$ 이 나타내는 영역부터 원하는 순서대로 $(X_N Y_M)$ 이 나타내는 영역까지 실행한다. 상기의 과정을 모두 실행하면 암호화된 하나의 텍스트 파일이 생성되고 이 텍스트 파일은 RGB 의 각 자릿수 마다 8 개씩, 총 24 개가 만들어진다.

3.3 암호화된 텍스트 데이터 전송 및 복호화 과정

24 개의 텍스트 파일을 전송할 때에는 암호화한 순서대로 보내는 것이 아니라 텍스트 데이터 마다 데이터 번호를 나타내는 태그를 추가한 뒤 순서를 무작위로 섞어서 보낸다. 데이터를 받는 사용자는 노노그램을 사용해서 각 텍스트 파일로부터 24 개의 비트맵을 만들고 생성된 비트맵들의 배치 순서를 암호화의 두 번째 Key 로 사용해서 각 텍스트 데이터의 태그와 맞게 배열함으로써 원본 이미지와 같은 결과를 만들어낸다.

그림 4. 원본 이미지와 잘못 복원된 이미지

4. 결과 및 토론

본 논문에서는 RGB 로 구성된 이미지 데이터를 노노그램을 사용해서 텍스트 데이터로 변환시킴으로써 이미지 암호화의 효과를 얻는 방안을 소개하였다. 상기에 제시된 방안은 노노그램과 image-to-text 기법의 특성으로 인해서 기존의 알고리즘과 대비되는 의의를 가지기도 하지만 그 만큼 여러가지 한계도 가지고 있다.

각 텍스트 데이터를 비트맵으로 변환하는 과정은 첫 번째 Key 인 $X_1, X_2, \ldots, X_N$, $Y_1, Y_2, \ldots, Y_M$ 값이 없으면 텍스트 데이터의 해석이 불가능하고 각 텍스트 데이터를 해석해서 24 개의 올바른 비트맵을 만들어냈다고 해도, 두 번째 Key 인 비트맵의 배치 순서를 모른다면 원본 이미지를 찾아내기 위해서는 비트맵을 배치하는 24!(약 2^{79})가지의 경우의 수를 모두 실험해봐야 한다. 이는 이중 암호화를 사용함으로써 원본 데이터가 제 3 자에의해 복화화될 가능성을 크게 줄이는 효과를 가져온다. 또한 기존의 image-to-image 로 암호화 방법과 달리 이미지 데이터를 텍스트 데이터로 암호화 함으로써 암호화된 데이터가 이미지로 복호화된다는 것을 은폐하여 제 3 자가 암호화된 데이터를 이미지 데이터로 복원하려는 발상을 사전에 차단할 수 있다는 것에 의의를 둘 수 있다.

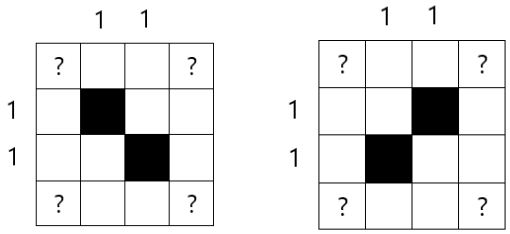

그림 5. 노노그램 중복 답안 예시. 물음표는 어떤 값이라도 상관 없다

본 논문에서 제시하는 알고리즘의 한계점은 여러가지가 존재한다. 첫째, 본 알고리즘은 RGB 로 구성된 이미지 데이터에만 적용한 방법이라는 점으로, 다른 형식으로 구성된 이미지 파일을 암호화 하려면 또 다른 알고리즘을 구상해야 한다. 두번째, 1 픽셀에 8bit 로 구성된 이미지 데이터를 양의 정수로 구성된 텍스트 데이터로 암호화 하기 때문에 암호화된 텍스트 데이터가 원본 이미지 데이터의 크기보다 커진다. 세번째, 노노그램을 사용해서 텍스트 데이터를 복원하는 과정에서 그림 5 와 같이 중복되는 답이 생길 수 있기 때문에[4] 항상 원본 이미지와 완벽하게 일치하게 복원하지는 못 할 가능성이 존재한다. 네번째, 기존의 알고리즘을 사용해서 25 x 25 픽셀 크기의 노노그램 퍼즐을 1000 개 풀기 위해 약 30 분이 필요하기 때문에[5] 데이터를 복호화 하는데 시간이 매우 많이 필요하다. 마지막으로 그림 4 와 같이 올바른 첫번째 Key 를 사용해서 복원한 24 개의 비트맵을 잘못 배치해도 원본이미지의 대략적인 모양을 알 수 있다는

한계점이 있어 추후 연구에서는 이를 개선하기 위한 연구를 진행해 나갈 예정이다.

사사의 글

이 연구는 한국연구재단의 중견연구자지원사업 (2019R1A2C2002449), 및 과학기술인문융합 연구사업 (2017M3C1B6070980)의 지원을 받아 수행되었음.

참고 문헌

1. Chiung-Hsueh Yu, Hui-Lung Lee, Ling-Hwei Chen, an efficient algorithm for solving nonograms, Springer Science+Business Media, LLC 2009.
2. Zhi-Hoing Guan, Fangjun Huang, Whnjie Guan, Chaos-based image encryption algorithm, Physics Letters A 346 (2005) 153--157.
3. Majid Babaei, a novel text and image encryption method based on chaos theory and DNA computing, Springer Science+Business Media B.V., 2012.
4. Daniel Berend, Dolev Pomeranz, Ronen Rabani, Ben Raziel, Nonograms: Combinatorial questions and algorithms, Discrete Applied Mathematics 169 (2014) 30--42.
5. Kuo-Chan Huang, Jia-Jun Yeh, Wei-Chiao Huang, Yan_Rong Guo, Lung-Pin Chen, Exploring effects of fully probing sequence on solving nonogram puzzles, ICGA Journal 40 (2018) 397--405